Atom Smashing using Machine Learning at CERN

Siddha Ganju
Master in Computational Data Science
Carnegie Mellon University

March 28-31, San Jose, CA
About Me

@SiddhaGanju

http://sidgan.me/

sganju1[at]cs[dot]cmu[dot]edu
Evaluation of Apache Spark as Analytics Framework for CERN’s Big Data Analytics

• Thanks to my mentors
  • Valentin Kuznetsov, Cornell University
  • Tony Wildish, Princeton University
  • Manuel Martin Marquez, CERN
  • Antonio Romero Marin, CERN
Project Scope

• Understanding CMS BIG data
  - Static + Streaming

• Exploring Apache Spark
  - Potential framework for big data analysis
  - Predict popular datasets in near real time
  - Facilitate Dynamic Data Placement
  - Efficient resource utilization
Outline

- CERN
- Understanding CMS Data
- Evaluation of Apache Spark
- Results and Conclusion
European Organization for Nuclear Research

LHC@CERN

Source: Manuel Martin Marquez, CERN
European Organization for Nuclear Research

LHC@CERN
Atom Smashing at CERN

• Compact Muon Solenoid (CMS) hunts for Higgs boson particles and clues to the nature of dark matter

• Beams of protons collide with energy of 13 TeV
  • Visualized in 3 dimensions
  • Digital summary of collisions

• Peta Byte (PB) -order data obtained, post filtering
CERN Computing Grid

- Recording
- Reconstruction
- Distribution

Source: Manuel Martin Marquez, CERN
Physics Workflow and Data Analytics

- Physics data
  - Different streaming data scenario
  - The new data is of course interesting!
  - Old data is all the more interesting!!
    - Re-processed old data with signals and background check influences the importance of new data
# Big Data Approach

<table>
<thead>
<tr>
<th>Volume</th>
<th>Velocity</th>
<th>Variety</th>
<th>Veracity</th>
</tr>
</thead>
</table>
| • Number of collisions in the LHC  
  • 15 PB order post filtering | • 300 Mb/s production rate  
  • Quick availability of results  
  • Real time feedback is NOT required | • Irregular data  
  • Different kinds data sets | • Uncertain  
  • Limited data integrity |

3/30/16  
Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
Data Acquisition – Structured sources

Worldwide LHC Computing Grid

PhEDEx: Physics Experiment Data Export
CMS Data Management System
CMS Workload Management System
PopDB
SiteDB
Dashboard

Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
Data Acquisition – Unstructured sources

• CERN Indico
• CERN Zenodo
• CERN Vidyo
Data is important
Data is really important

Source: Tony Wildish, Princeton University
Data is indeed important

The LHC collects about 25 million gigabytes of data per year

*Binary data
Note: All numbers are approximate.
Source: “Particle Physics Tames Big Data,” Leah Hesla, Symmetry, 1 August 2012
Outline

- CERN
- Understanding CMS Data
- Evaluation of Apache Spark
- Results & Conclusion
Dynamic Data Placement

- Efficient resource utilization
- Reduce redundancy

DCAFPilot

- Does not learn

- Machine learning
- Identify perfect model for popularity

scikit learn
machine learning in Python

Spark

- Real time
- RDD level parallelism

Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
Data and Computing Analysis Framework

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAF Pilot/doc/talks/Pilot1/images/DCAF Pilot_flow.png
1K popular
&
10K unpopular
78 features
25 relevant features
## Feature Extraction

<table>
<thead>
<tr>
<th>id</th>
<th>cpu</th>
<th>nllumis</th>
<th>nfiles</th>
<th>nblk</th>
</tr>
</thead>
<tbody>
<tr>
<td>creator</td>
<td>tier</td>
<td>nevt</td>
<td>proc Evts</td>
<td>nusers</td>
</tr>
<tr>
<td>wct</td>
<td>size</td>
<td>rnaccess</td>
<td>primds</td>
<td>nsites</td>
</tr>
<tr>
<td>totcpu</td>
<td>rtotcpu</td>
<td>rnusers</td>
<td>parent</td>
<td>nrel</td>
</tr>
<tr>
<td>naccess</td>
<td>era</td>
<td>dtype</td>
<td>dbs</td>
<td>dataset</td>
</tr>
</tbody>
</table>
Data and Computing Analysis Framework

![Diagram of Data and Computing Analysis Framework](https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/DCAFPilot_flow.png)

Transforming to classification problem

• Classification problem
  • Target column is 1(popular) when naccess is greater than 10 and nusers is greater than 5

```python
def convert(df):
    threshold_naccess = 10
    threshold_nusers = 5
    return df['naccess'] > threshold_naccess and df['nusers'] > threshold_nusers
```
Frequency of Accesses

Naccess cut found by plotting the entire 2014 dataset against frequency of access
600KB compressed data frames
600KB compressed/file
* 52 files/year
* 3 years/run
~15 PB/year
Spark

RDD Objects

Spark Client (Application Master)

Task Scheduler

Worker

rdd1.join(rdd2)
  .groupBy(...)
  .filter(...)

Scheduler and RDD Graph

Cluster Manager

Threads

Block Manager

BlockInfo

MemoryStore

DiskStore

ShuffleBlockManager

Source: 15-319 / 15-619 Cloud Computing, CMU
Apache Spark analysis

• Fast and compatible with already existing HDFS at CERN
• Run in Hadoop clusters through MESOS or Spark's standalone mode
  • Process data in HDFS
  • Batching of a week’s data
  • Processing for new workloads like streaming (live prediction of each week), interactive queries, and machine learning.
Rolling Forecast

1. New data
2. Retrain model
3. Make prediction

2013.csv
Row 1
Row 2
Row 3
Row 4
Row 5
Row 6

Machine Learning

Model

20140101-20140107
Row 1
Row 2
Row 3

Prediction
1 Popular
0 Unpopular
0 Unpopular
0 Unpopular
0 Unpopular
1 Popular
1 Popular
1 Popular

Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
Rolling Forecast

New data

Retrain model

Make prediction

Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
Outline

- CERN
- Understanding CMS Data
- Evaluation of Apache Spark
- Results & Conclusion
• Parallel analysis on distributed data
• Decreased total execution time for the individual algorithm or combined ensemble to execute
Popularity Metrics

Popularity of Random datasets

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPlot/doc/talks/Plot1/images/popularity.jpg
Popularity Metrics - nusers

Total number of users who accessed the dataset

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAF Pilot/doc/talks/Pilot1/images/nusers_cloud.gif
Popularity Metrics - totcpu

Total CPU hours spent analyzing a dataset

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPlilot/doc/talks/Pilot1/images/cpu_cloud.gif
Popularity Metrics - naccess

Count of individual accesses to a dataset

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAF Pilot/doc/talks/Pilot1/images/naccess_cloud.gif
CPU Usage

• Using psutil (python system and process utilities)
  • /proc/pid/stat of pid

• Approximately 25% of the CPU is always being used by the algorithms
CPU Usage

• Importance
  • Lots of processes submitted to the CERN Grid
  • All processes should work in a cooperative environment
  • All processes should be able to access the required resources

• Statistical difference between CPU usage
  • Scikit-learn uses more CPU for longer time duration
Naïve Bayes – Apache Spark
Stochastic Gradient Descent – Apache Spark
## Performance matrix – Apache Spark

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Memory Used (GB)</th>
<th>Execution Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>5.072920</td>
<td>8184.136</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>5.376336</td>
<td>1204.97</td>
</tr>
<tr>
<td>Stochastic Gradient Descent</td>
<td>5.181784</td>
<td>1261.94</td>
</tr>
</tbody>
</table>

Siddha Ganju, CMU, CERN; Strata+Hadoop 2016
RAM Usage

• Incremental learning
  • With each iteration more RAM is utilized
  • Amount of data being processed is continually increased

• Graph does not indicate memory leak
  • Rolling approach
  • Incremental learning part
RAM Usage

• Statistical difference between RAM usage
  • Scikit-learn uses more RAM for longer time duration
  • RDD parallelism in Apache Spark
Naïve Bayes – Apache Spark
Random Forest – Apache Spark
Stochastic Gradient Descent – Apache Spark
Results

• False Positive Rate
  • FPR = FP / (FP + TN)

• False positive
  • Unnecessary replication
  • Dataset unpopular but predicted popular

• True Negative
  • Correct prediction
  • Dataset unpopular and predicted unpopular
Results

• Difference is not statistically significant

High precision and high recall
Returns many correctly labelled results

Returns many results
Most predictions incorrect

Returns few results
Predicted labels are correct
Results

• Accuracy problems because of data
  • Not reliable
  • Data can be unbalanced
  • Data Integrity

• Using the cross validation scoring method

```python
from sklearn.metrics import accuracy_score
score(clf, test, target_test)
```
### Confusion matrices

<table>
<thead>
<tr>
<th>RF</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>570374</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>6438</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SGD</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>568040</td>
<td>3351</td>
</tr>
<tr>
<td>F</td>
<td>2336</td>
<td>3042</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NB</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>568415</td>
<td>6</td>
</tr>
<tr>
<td>F</td>
<td>12783</td>
<td>6432</td>
</tr>
</tbody>
</table>
Confusion matrix

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>.98</td>
<td>.86</td>
<td>.98</td>
<td>.92</td>
</tr>
<tr>
<td>SGD</td>
<td>.96</td>
<td>.98</td>
<td>.62</td>
<td>.76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>.99</td>
<td>.99</td>
<td>1</td>
<td>.99</td>
</tr>
<tr>
<td>SGD</td>
<td>.99</td>
<td>.99</td>
<td>.99</td>
<td>.99</td>
</tr>
</tbody>
</table>
From history to real

- Historical study of popularity & prediction of future popularity are two different concepts
- What was popular in the past
  - naccess, totcpu, nusers, likely to be related to popularity
- Predictions
  - Don't know the number of accesses
    - Use different features
    - From solid understanding (history) to future prediction is difficult
    - May not be able to use the best metrics
Apache Spark v/s scikit learn

- Real time results
- Mesos Cluster manager
- Interfaced with Hadoop Distributed File System (HDFS)
- Aggressive caching in memory
- Faster and Scalable
- RDD level parallelism
- Version Used:
  - Release 1.4.0

- Works directly as a library
- User-friendly
- Benchmarked models already in use
- Deployed over:
  - Python 2.7.5
  - Scikit version 0.16.0
Python scikit-learn

• High CPU usage 😞
• High RAM usage 😞
• More Time 😞
Apache Spark

• CPU usage looks Good 😊
• RAM usage looks Good 😊
• Less Time 😊
Thank You
References

1. Siddha Ganju, 2015, Evaluation of Apache Spark as Analytics as framework for CERN's Big Data Analytics, CERN Zenodo network
2. Toby Segaran, 2007, Programming Collective Intelligence, O’ Reilly
3. Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills, Advanced Analytics with Spark, Patterns for Learning from Data at Scale, O’ Reilly
4. Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia, Learning Spark, Lightning-Fast Big Data Analysis, O'Reilly