Hepdoop:

High-Energy Physics Analysis using Hadoop

Wahid Bhimji, Tim Bristow, Andrew Washbrook

SUPA – School of Physics and Astronomy, University of Edinburgh, Edinburgh, United Kingdom

We perform a LHC data analysis workflow using tools and data formats that are commonly used in the "Big Data" community outside HEP. These include Apache Avro for serialisation to binary files, Pig and Hadoop for mass data processing and Python Scikit-Learn for multi-variate analysis. Comparison is made with the same analysis performed with current HEP tools in ROOT.

Introduction

"Big Data" is now business-as-usual in the private sector including big names such as Twitter, Facebook, Amazon and Google. High-Energy Physics (HEP) is often cited as the archetypal "Big Data" use case, however it currently shares very little of the toolkit used in the private sector which is dominated by the Apache Hadoop "ecosystem". We use those tools to perform aspects of a HEP analysis as part of a longer term goal of interaction between big data communities.

For this study we use established tools with large user communities (see box alongside). We aim for performance that is good-enough (i.e. of the same order as HEP tools rather than seeking to exceed it) and make ease-of-use also an important consideration.

Technologies used

Hadoop is a framework "for the distributed processing of large data sets across clusters of computers using simple programming models"[1]. It provides easy execution of MapReduce programs and the HDFS distributed filesystem.

Pig produces MapReduce programs for Hadoop using a query language called Pig Latin which offers [2] "ease of programming" for parallel execution of tasks, "optimisation opportunities" as task execution is optimised automatically and "Extensibility" through User Defined Functions (UDF)s.

Avro [3] is a file-based binary-format data serialization system. It relies on schemas but these are stored in the file which are thus self-describing. Files are compressed, with a choice of codecs, and tools for reading/writing are available in several languages including Pig.

Scikit-Learn [4] is a mature, well-documented and fully-featured machine learning package for python built on NumPy and matplotlib. The later are also used for visualisation and histogramming in this project.

ROOT [5] is the most commonly used data analysis technology in HEP both for data structures and the analysis toolkit. A number of tools have been developed for ROOT including **TMVA** [6] for multi-variate analysis.

HEP Analysis Use-case

Analysis stages are given below with generic titles and HEP terms.

Data Filtering: Skimming/Slimming → Data Mining: Cut-optimisation / MVA → Data Visualisation: Histogramming

Data Filtering

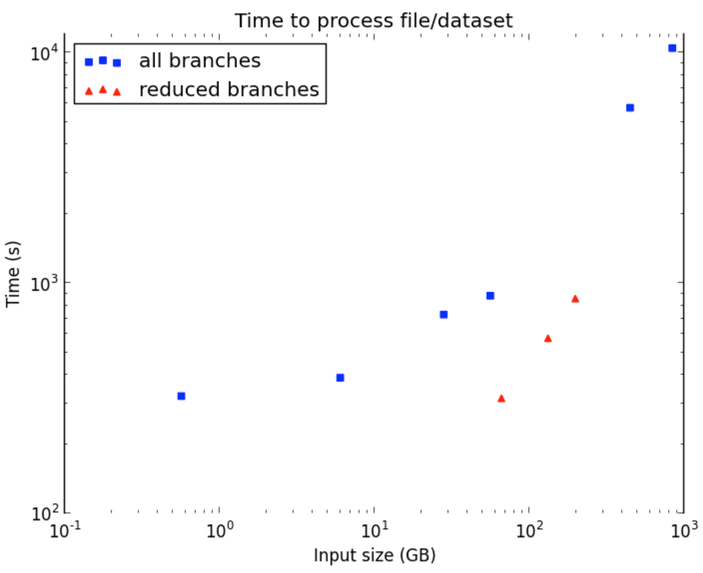
We use data from the *ATLAS* experiment in the popular centrally-produced *SMWZ_D3PD* format which contains ROOT TTrees with 5860 branches of simple types and vectors (~65 KB/event). Due to the large file sizes, an analysis group may conduct a "Filtering" stage. The output is in the same structure but with a selection of events and reduction of branches. We use ~1TB of input data which is converted into zlib-compressed Avro. We apply a selection motivated by an analysis of Higgs Boson decays to two b-quarks (H to bb) which has an event selection efficiency of 5% and selects 109 branches.

```
REGISTER 'MySelections.py' using jython as myfuncs;
DEFINE  AvroStorage
org.apache.pig.piggybank.storage.avro.AvroStorage;
input = LOAD '$input' USING AvroStorage();
data = FOREACH input GENERATE $branches;
outputdata = FILTER data BY MET_RefFinal_et > 45.0
AND  myfuncs.passesJetSelection(jet_AntiKt4TopoEM)
AND ...
STORE outputdata INTO './results' USING AvroStorage();
```

Illustrative Pig code is shown to the left. Filtering is done with simple comparisons and more complex selections via a python UDF. For the latter, the same code can be reused in both Pig and ROOT.

Performance

The Hadoop processing time versus the size of the input dataset, when run on a 500 core Hadoop installation at CERN, is shown to the right. The ROOT-based tool centrally available on ATLAS for this purpose (*filter-and-merge*) has similar 270s run time on a single 760 MB file.



However with custom ROOT code, reading only the branches used, we were able to achieve a single-file run time of 30s and the application could be run in parallel on a cluster of this sort, with a batch system or using PROOF, possibly with better scaling than seen here for Hadoop.

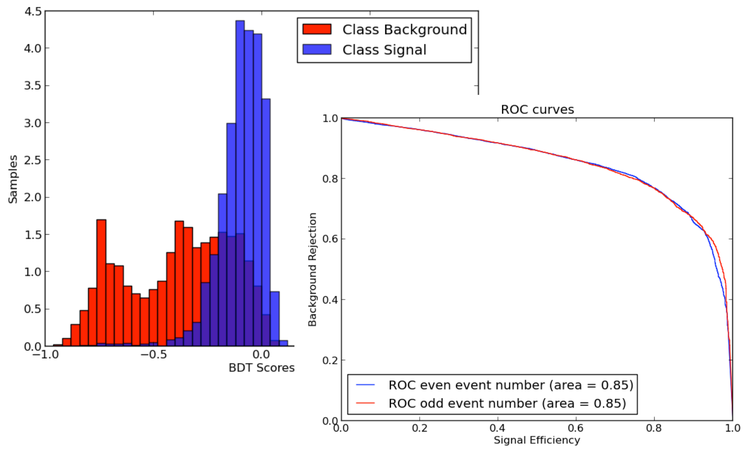
We do also observe, however, an improvement of performance in Hadoop/Avro when some large, unused, vector branches are removed (leaving 2300 "branches" at 2.5KB/event) (red triangles above).

Data Mining

Filtered data is then "mined" by applying further linear selections (which we implement in Pig) or a Multi-Variate Analysis (MVA) such as a Boosted Decision Tree (BDT) (which we implement in scikit-learn). This is run on the complete input data for the ATLAS H to bb analysis.

Illustrative code is shown here: *trainingData*, *trainingClasses* and *testingData* are NumPy arrays that can either be read from ROOT TTrees (using rootpy [7]) or filled from Avro.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
ada =
AdaBoostClassifier(DecisionTreeClassifier(max_depth=4,
compute_importances=True,min_samples_split=2,min_s
amples_leaf=100),n_estimators=400, learning_rate=0.5,
algorithm="SAMME",compute_importances=True)
ada.fit(trainingData, trainingClasses)
return ada.decision_function(testingData)
```



Using IPython[8] to enable parallel training on samples, execution time in Scikit is similar to TMVA.

Outputs of the BDT show the same results as with ROOT/TMVA - but this package also allows for alternative approaches.

Conclusion

Key components of a HEP analysis have been run using out-of-the-box "Big data" tools on Hadoop. This enables use of wider resources such as Amazon Elastic Map Reduce and can benefit from large user and developer communities as well as further off-the-shelf tools for use in, for example, provisioning, monitoring and scheduling.

The performance seen here for filtering is inferior to the ROOT-based analysis, particularly in the case of large complex nested structures. To address this issue, we are currently exploring the use of columnar storage formats such as Parquet [9].

References

[1] http://hadoop.apache.org/
[2] http://pig.apache.org/
[3] http://avro.apache.org/
[4] http://scikit-learn.org/
[5] http://root.cern.ch/
[6] http://tmva.sourceforge.net/
[7] http://www.rootpy.org/
[8] http://ipython.org/
[9] http://parquet.io/

Acknowledgements

Thanks to the CERN IT-DSS group for use of Hadoop resources.