Big Data Helps Particle Physicists to Concentrate on Science

Saba Sehrish
Fermi National Accelerator Laboratory
Batavia, IL 60510
ssehrish@fnal.gov

Jim Kowalkowski
Fermi National Accelerator Laboratory
Batavia, IL 60510
jbk@fnal.gov

Oliver Gutsche
Fermi National Accelerator Laboratory
Batavia, IL 60510
gutsche@fnal.gov

Matteo Cremonesi
Fermi National Accelerator Laboratory
Batavia, IL 60510
matteoc@fnal.gov

Alexey Svyatovskiiy
Princeton University
Princeton, NJ 08544
alexeys@princeton.edu

Jim Pivarski
Princeton University
Princeton, NJ 08544
pivarski@princeton.edu

1. SUMMARY

In this poster, we evaluate Apache Spark for High Energy Physics (HEP) analyses. Our goal is to understand how well this technology performs for HEP-like analyses in both HPC and Hadoop ecosystem. We use an example from the Compact Muon Solenoid (CMS) experiment [4] at the Large Hadron Collider (LHC) in Geneva, Switzerland, the highest energy particle collider in the world. The CMS detector measures different properties of the particles produced in a collision, such as tracks left by charged particles and energy deposits from all particles that interact via photons and gluons. Our use case focuses on searching for new types of elementary particles explaining Dark Matter in the universe. In particular, this search is looking for a signature in the events commonly referred to as mono-X where X can be a light quark or gluon, a vector boson, or a heavy quark such as a bottom or top quark. We focus our search on the monoTop signature, where the detectable particle is a single, unbalanced top quark.

The structured event data is translated into a flat n-tuple; each row of a table represents an event, different particles in an event (photons, electrons, taus) and their properties (pt, eta, phi). Often, the n-tuples are still too big for interactive analysis (2 TB). Therefore, the contents and the number of events are reduced (GBs). Eventually, quantities from the final n-tuple are aggregated and plotted as histograms. The time scale of the complete Dark Matter workflow can range from days to weeks, depending on the number of events needed for analysis.

The traditional user analysis workflow for CMS data uses two C++ frameworks: CMSSW, specially designed for analyzing CMS data, and ROOT [3], which is a general, experiment-independent C++ toolkit. The ROOT framework provides statistical tools and a serialization format to persist reconstructed and transformed objects in files. We provide two implementations of this analysis workflow using Spark [1, 6].

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Spark on Hadoop: We developed a library to convert the official experiment data files in ROOT to Apache Avro row-based format readable by Spark. The skimming and slimming on the input data is implemented by using Spark's map, flatMap and filter transformations. We use Apache Parquet columnar format to store the intermediate results between stages of the calculation in HDFS, allowing to easily ingest data into Dataframes and Datasets.

Spark on HPC [2]: We wrote a converter in Python to convert the official experiment data files in ROOT to the HDF5 format. HDF5 [5] is well supported at HPC platforms including NERSC, and will allow us to use other HPC programming interfaces (MPI). Our approach is to convert each ROOT TBranch representing a particle in an event to a group in HDF5, and each leaf representing a property with in a branch to 1D dataset in HDF5. We stored the highly structured data into flat tables and column-oriented structure to allow distributed processing across groups as needed. We implemented a customized HDF5 reader in Spark/Scala to read in an HDF5 group with specified datasets into a Spark DataFrame. The data partition is based on the number of elements in each HDF5 dataset per group across all the input files. we use HDF5 hyper slabs to read in chunks to allow maximal parallelism while reading data into DataFrames. We implemented skimming and slimming code in Scala using filters, UDF and SQL queries on Spark DataFrames.

We ran the following test on each platform: The input data was 170 GB represented in ROOT and Avro. HDF5 used 46 GB because only the columns needed in this analysis were converted and compressed. On Edison, we used 7 nodes, 20 cores and executor memory of 58GB. After caching, it took about 2.0 seconds to calculate the sum of weights. The same test on Princeton big data cluster took 14 seconds. The sample data set we used has 3.7 million events; it took 1.7 seconds to count the number of events. On Edison, we generated 7 output files, one per particle and two for the event info. Writing all the DataFrames after applying cuts took couple of minutes (< 6).

Spark is relatively new and emerging technology, and its use, especially in the HEP community, is in exploratory stages. The learning curve involved with the use of this new technology, especially using Scala, cannot be ignored. However, the availability of APIs in R and Python improves the beginner user experience. Other advantages include task distribution and user controlled data partitioning. We have seen good scaling behavior of Spark applications with increase in dataset size and the number of nodes with no extra work. Encoding skimming workflow using Scala best practices is challenging along with optimal use of Spark DataFrame features. The documentation and error reporting should be improved. However, the ease of use, reasonable performance and good scalability makes Spark a viable candidate for our future work.

2. ADDITIONAL AUTHORS

Cristina Mantilla, Fermi National Accelerator Laboratory, email: `cmantill@fnal.gov` and Bo Jayatilaka Fermi National Accelerator Laboratory, email: `boj@fnal.gov`

3. REFERENCES

[1] Spark. https://spark.apache.org.

[2] Spark Distributed Analytics Framework at NERSC. https://www.nersc.gov/users/data-analytics/data-analytics/spark-distributed-analytic-framework.

[3] R. Brun and F. Rademakers. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth.*, A389:81–86, 1997.

[4] S. Chatrchyan et al. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008.

[5] The HDF Group. Hierarchical Data Format, version 5, 1997-2016. /HDF5/.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.