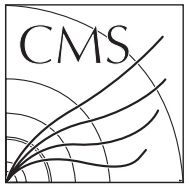
Available on CMS information server

CMS CR -2018/318



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

28 October 2018

Using Big Data Technologies for HEP analysis

Matteo Cremonesi for the CMS Collaboration

Abstract

The HEP community is approaching an era were the excellent performances of the particle accelerators in delivering collision at high rate will force the experiments to record a large amount of information. The growing size of the datasets could potentially become a limiting factor in the capability to produce scientific results timely and efficiently.Recently, new technologies and new approaches have been developed in industry to answer to the necessity to retrieve information as quick as possible by analyzing PB and EB datasets. Providing the scientists with more modern computing tools will lead to rethinking the principles of data analysis in HEP, making the overall scientific process faster and smoother. In this talk, we are presenting the latest developments and the most recent results on the usage of Apache Spark for HEP analysis. The study aims at evaluating the efficiency of the application of the new tools both quantitatively, by measuring the performances, and qualitatively, focusing on the user experience. The first goal is achieved by developing a data reduction facility working together with CERN Openlab and Intel, CMS replicates a real physics search using Spark-based technologies, with the ambition of reducing 1 PB of public data collected by the CMS experiment to 1 TB of data in a format suitable for physics analysis in 5 hours. The second goal is achieved by implementing multiple physics use-cases in Apache Spark using in input preprocessed datasets derived from official CMS data and simulation. By performing different end-analyses up to the publication plots on different hardware, feasibility, usability and portability are compared to the ones of a traditional ROOT-based workflow.

Presented at *CHEP 2018 Computing in High-Energy Physics 2018*

Using Big Data Technologies for HEP Analysis

Oliver Gutsche[2], Luca Canali[1], Illia Cremer[4], Matteo Cremonesi[2], Vasileios Dimakopoulos[1], Peter Elmer[5], Ian Fisk[3], Maria Girone[1], Siew-Yan Hoh[7], Bo Jayatilaka[2], Jim Kowalkowski[2], Viktor Khristenko[1], Andrew Melo[6], Evangelos Motesnitsalis[1], Jacopo Pazzini[7], Jim Pivarski[5], Saba Sehrish[2], Kacper Surdy[1], Alexey Svyatkovskiy[5], Marco Zanetti[7]

[1]European Organization for Nuclear Research CERN, Geneva, Switzerland
[2]Fermi National Accelerator Laboratory, Batavia, IL, USA
[3]Flatiron Institute of the Sions Foundation, New York, NY, USA
[4]Intel Corp.
[5]Princeton University, Princeton, NJ, USA
[6]Vanderbilt University, Nashville, TN, USA
[7]University of Padova, Padova, Italy

E-mail: matteoc@fnal.gov

Abstract. The HEP community is approaching an era were the excellent performances of the particle accelerators in delivering collision at high rate will force the experiments to record a large amount of information. The growing size of the datasets could potentially become a limiting factor in the capability to produce scientific results timely and efficiently. Recently, new technologies and new approaches have been developed in industry to answer to the necessity to retrieve information as quick as possible by analyzing PB and EB datasets. Providing the scientists with more modern computing tools will lead to rethinking the principles of data analysis in HEP, making the overall scientific process faster and smoother.

In this talk, we are presenting the latest developments and the most recent results on the usage of Apache Spark for HEP analysis. The study aims at evaluating the efficiency of the application of the new tools both quantitatively, by measuring the performances, and qualitatively, focusing on the user experience. The first goal is achieved by developing a data reduction facility: working together with CERN Openlab and Intel, CMS replicates a real physics search using Spark-based technologies, with the ambition of reducing 1 PB of public data collected by the CMS experiment to 1 TB of data in a format suitable for physics analysis in 5 hours.

The second goal is achieved by implementing multiple physics use-cases in Apache Spark using in input preprocessed datasets derived from official CMS data and simulation. By performing different end-analyses up to the publication plots on different hardware, feasibility, usability and portability are compared to the ones of a traditional ROOT-based workflow.

1. Introduction

The scientific method is based on comparing predictions to experimental data, in order to confirm or disprove new theories. In high energy physics (HEP), such data are collected by experiments that can detect fundamental particles once they are produced in the collision of beams provided by accelerators like the LHC at CERN.

Particle detection is an extremely complicated process. It consists in recording the physics quantities (like energy or flight path) of the particles generated in a collision. Such quantities are measured by the interaction of the particles with the different components of the detector used to perform the experiment. Given the complexity of the detector design, this process involves performing almost one hundred million independent measurements. All the measurements that happen in a single collision are collectively called event.

The event is the unit which the whole HEP analysis process is based on. Event by event, detector signals (as well as simulated signals) must be converted into the physics properties associated to the particles that produced them. Complex algorithms are applied in order to reconstruct such information. This step is computationally expensive and it is usually organized centrally by each experiment, in order to make the best use of the available resources and to best serve the need of the researchers and the priorities of the experiment. The reconstructed events are provided to the collaboration of physicists in a shared format that is the input to the final analysis.

Usually the total size of the datasets as provided by central processing is too large to allow for interactive analysis. Researchers or groups of researchers exploring similar physics questions rely on several steps of data processing, filtering unnecessary events and eliminating unnecessary variables from the original datasets to get a manageable sub-sample. The optimization of this process is left to the individual.

In the next years the experiments at CERN will face a substantial increase in the data production rate due to a planned major upgrade of the accelerators. In order to ensure continuity in the production of high quality scientific results, the inefficiencies that are affecting the current analysis approach must be trimmed. The need to investigate alternative possibilities to perform physics analysis in a more efficient way is therefore becoming imperative.

Recently, new toolkits and systems have emerged outside of the HEP community to analyze Petabyte and Exabyte datasets in industry, collectively called "Big Data." These new technologies use different approaches and promise a fresh look at analysis of extremely large datasets.

In this paper, we focus on the application of Apache Spark [1] to the HEP analysis problem. We incorporate lessons learned from our previous investigations [2] as well as new tools developed to enable HEP analysis in Apache Spark. Scalability and usability studies are performed and the latest findings are presented.

2. The Traditional Analysis Workflow and Its Limitations

The traditional HEP analysis workflow is based on the usage of the ROOT framework [4], a general, experiment-independent C++ toolkit. It provides statistical tools and a serialization format to persist reconstructed and transformed objects in files.

The centrally-produced datasets are provided in ROOT format, with a file-based data representation and a class structure with branches. The data management systems do not allow to extract branches efficiently from nested ROOT files, therefore physicists set up workflows that involve several steps of data processing, each one of them staging out intermediate outputs.

A first step of ntupling is performed in order to modify the event content. Immutable branches are duplicated in a disk-to-disk copy that contains new needed branches, while unused ones are removed. At this stage, the information is selected to serve a smaller group of researchers performing similar measurements or searches. Although the total size of the output is smaller, it is still too big to allow for interactive analysis.

A second step that involves dropping uninteresting events (skimming) or additional unused branches (slimming) is therefore necessary to limit latencies. The output is a disk-to-disk copy where the immutable information is once again duplicated, but the class structure is translated into a "flat" format, in which events are rows of a table with primitive numbers or arrays of

Table 1. The results of scalability tests for different input size

Input Size	Execution Time
22 TB	58 min
44 TB	83 min
66 TB	149 min
88 TB	180 min
110 TB	212 min

numbers as columns. At this stage, the information is usually selected to serve the scope of a single analysis. The size is reduced by an order of magnitude. Quantities from the final ntuples are aggregated and plotted as histograms.

These steps require the usage of grid and batch resources to exploit parallelization. Significant burden of tedious and time-consuming manual bookkeeping and failure re-submission is put on the individual analyst or analysis groups, that produces an inefficient job splitting, with suboptimal parallelization. This results in a convoluted approach that limits interactivity. The analysis frameworks that support such workflows are group- or analysis-specific, often hardware-specific, limiting the portability and stimulating the multiplication of individual codes with similar functionalities. The duplication of immutable branches that happens at each stage of the workflow brings significant usage of storage space, making such an approach not sustainable on the long run.

3. The Scalability Test

In a previous usability study [2] of Apache Spark, we implemented an analysis workflow by converting data into the AVRO [7] format and uploaded it to the HDFS [8] file system of our development cluster. The biggest impediment to use the new technology as identified by the analysts was the need to convert the data in the new format.

To enable Apache Spark to understand the data structures of the ROOT files more directly, we developed a library called spark-root [11]. It is based on a Java-only implementation of the ROOT I/O libraries which offers the ability to translate ROOT data structures into Spark DataFrames (DFs) and RDDs. This enables Apache Spark to read ROOT files directly from HDFS and also from the EOS [12] storage system through a new Hadoop-xrootd connector [13]. The EOS is an open source distributed disk storage system in production since 2011 at CERN. The Hadoop-XRootD Connector allows the communication between the EOS and Hadoop. Specifically, it is a Java library that access files directly through the xrootd protocol [14] without the need to import/export to HDFS.

In this paragraph, tests of performance, efficiency and scalability of the new tools are performed. A Spark workflow that reproduces a real physics measurement is ran on the analytix cluster at CERN. The infrastructure is comprised of almost 1300 cores and 7TB of RAM. The input is public data collected by the CMS experiment in 2011, stored as ROOT files on the EOS at CERN.

The test is performed by executing the workflow for different size of the input data, in order to understand how the execution time scales with input size. As a second step, the same workflow is executed for a specific input size while scaling up the available resources (executors/cores). The tests were repeated for two different instances of the EOS storage, namely EOS Public and EOS UAT, in order to identify if the network throughput and the storage infrastructure affect the performances.

Table 1 and Fig. 1 show a linear dependence between the input size and the execution time. The system is able to reduce 110 TBs in 212 minutes with no further optimization.

Figure 1. Performance for different input size

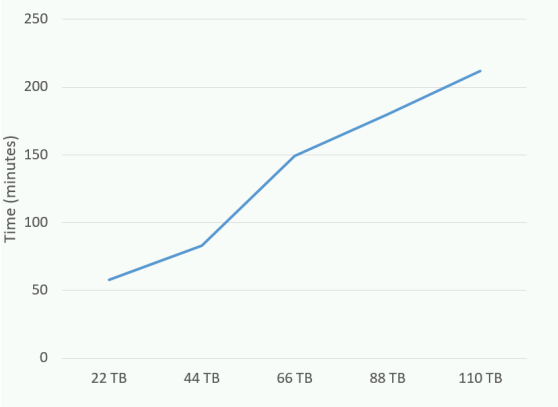


Table 2. The results from scaling up the memory

Number of Executors/Cores	Total Memory	Execution Time
74/148	0.5TB	81 min
148/296	1TB	53 min
222/444	1.5TB	52 min
296/592	2TB	51 min
370/740	2.5TB	50 min
444/888	3TB	50 min

Figure 2. Performance for different memory size

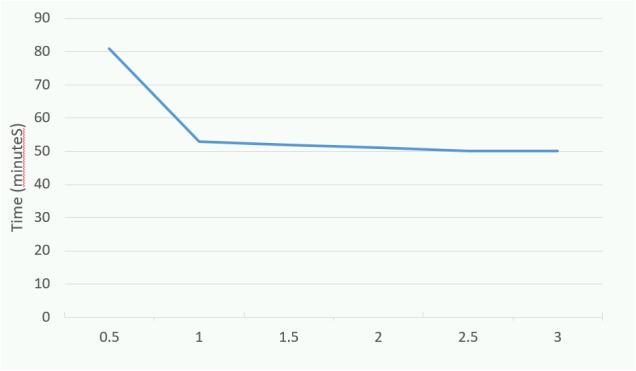


Table 2 and Fig. 2 show the results as a function of the allocated memory, obtained by adjusting the number of executors and fixing the memory per executor to 7GB. A plateau in the performance is reached at a specific memory value.

Table 3 and Fig. 3 show a similar trend when performances are tested scaling up the number of cores. This effect is caused by the saturation of the available network bandwidth. It is also evident when monitoring the total throughput of the network, as shown in Table 4 and Fig. 4.

The network does not seem to be able to fetch files fast enough to adapt to the scaling computing resources. The new resources are waiting to be used when the files are ready for processing, resulting in a network bound workflow. This effect has been investigated and a defect of the Hadoop-XRootD connector in reading files from EOS has been found. The issue

Table 3. The results from scaling up the number of cores

Number of Executors/Cores	EOS public	EOS UAT	HDFS
111/222	81 min	153 min	41 min
222/444	52 min	146 min	35 min
296/592	51 min	144 min	33 min
407/814	50 min	140 min	29 min

Figure 3. Performance for different number of cores

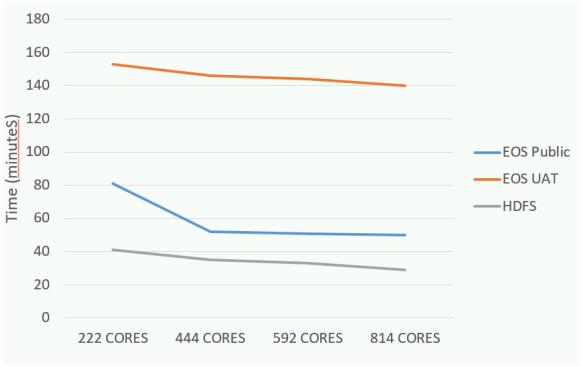


Table 4. Network throughput when scaling the resources

Cores	EOS public	EOS UAT
222	15GBytes/s	6GBytes/s
444	19GBytes/s	7.5GBytes/s
592	21GBytes/s	7.5GBytes/s
814	21GBytes/s	7.5GBytes/s

Figure 4. Network Throughput while scaling the resources

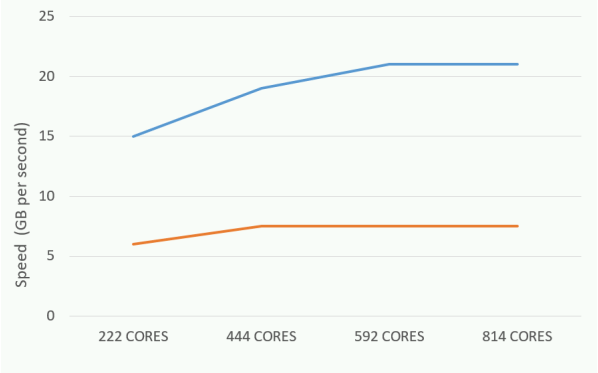


Table 5. Aggregate results of re-executing tests by scaling up the input data size on EOS public, EOS UAT

Input	EOS PUBLIC	EOS UAT
22 TB	16 mins	13 mins
44 TB	30 mins	25 mins
66 TB	38 mins	38 mins
88 TB	48 mins	57 mins
110 TB	56 mins	59 mins

Figure 5. Performance of the tests for different input size with new Hadoop-XRootD connector configuration on EOS public, UAT

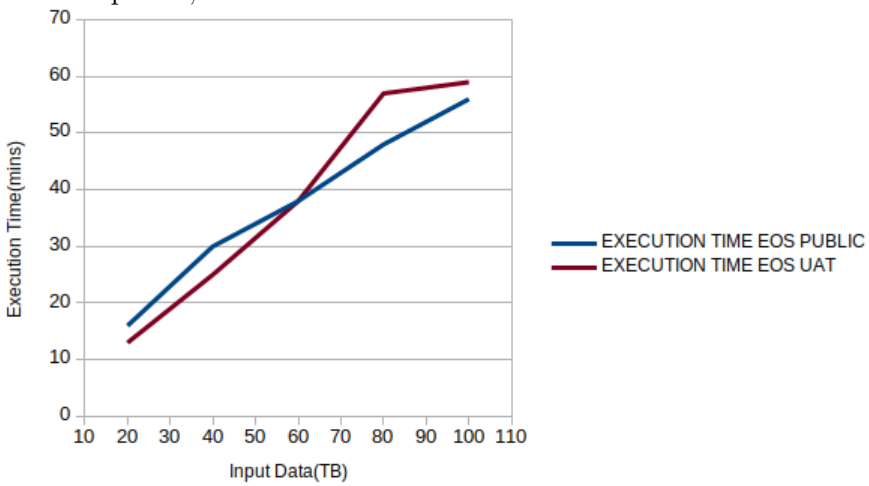


Table 6. Aggregate results of re-executing tests by scaling up the memory on EOS public, EOS UAT

Input	Memory	EOS PUBLIC	EOS UAT
22 TB	0.5 TB	32 mins	21 mins
22 TB	1 TB	28 mins	16 mins
22 TB	1.5 TB	16 mins	15 mins
22 TB	2 TB	16 mins	13 mins
22 TB	2.5 TB	16 mins	13 mins

was caused by the buffer size used to read the files from the storage service. It was set to 32MB and it resulted in an unnecesary collection of data from file, increasing the network throughput. The buffer size was decreased to 128 KB and all the tests re-executed for both the EOS public and the EOS UAT instances. The final results are summarized in Tables 5-7 and Figs. 5-7.

The dependences are linear as expected, with a smoother scaling and a smaller slope. The new results show a significant significant improvement, proving the capability to reduce 110 TB of data in less than an hour.

4. The Usability Study

The main goal of the usability test described in this section is testing the user experience, from the ability to setup and run a Spark-based analysis workflow to the portability of such workflow

Figure 6. Performance of the tests for different memory with new Hadoop-XRootD Connector configuration on EOS public, UAT

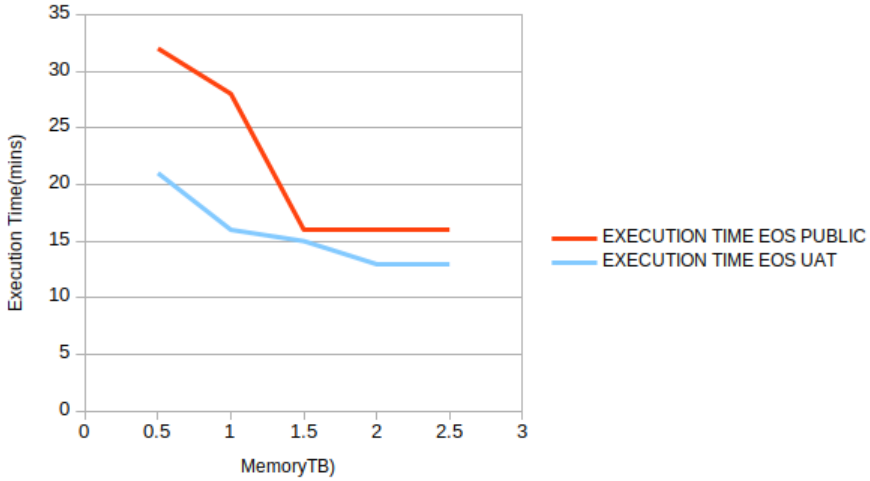
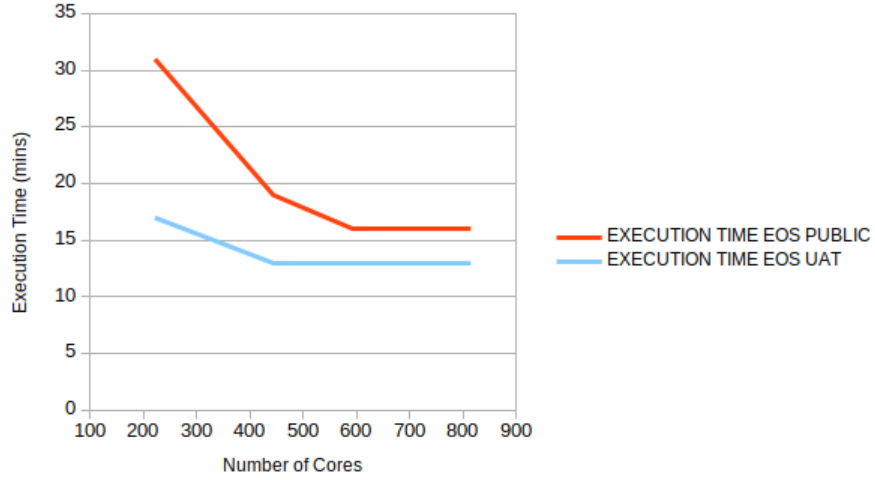


Table 7. Aggregate results of re-executing tests by scaling up the cores on EOS public, EOS UAT

Input	Cores	EOS PUBLIC	EOS UAT
22 TB	222	31 mins	17 mins
22 TB	444	19 mins	13 mins
22 TB	592	16 mins	13 mins
22 TB	814	16 mins	13 mins

Figure 7. Performance of the tests for different number of cores with new Hadoop-XRootD Connector configuration on EOS public, UAT



to different use cases and, most importantly, different hardware.

To perform this test, two similar workflows tuned to run on different clusters were ran. A Spark cluster at Vanderbilt University was used. It consists in 1000 cores with 5 TB of RAM. A second cluster hosted at the University of Padova, with 40 cores and 16 GB of RAM, was also employed in this test. The two workflows shared a similar structure: load standard ROOT files as Spark DFs making use of the spark-root library, open them over xrootd with the Hadoop-xrootd connector, use Spark to transform DFs and aggregate them into histograms with the Histogrammar [10] package, produce plots and tables from the histograms.

The first step of the test consisted in verifying how easily such workflow can be set up by a newcomer. A first year undergraduate student in computer science, with no physics knowledge and limited computing knowledge, approached the issue. Starting from scratch, he was able to learn the basic functionalities of the new tools and run the entire workflow in a day. The simplicity of the Spark workflow to be set up in a short timescale is a clear advantage when compared to the timescale that is needed for a newcomer once approaching the classical ROOT-based workflow.

The second step consisted in adapting to the Vanderbilt cluster the workflow tuned to run at the Padova cluster. The major showstopper encountered was the environment setup. This can be solved developing a shared library that generalizes the site configuration. Additional improvement is also required for the packaging of the Hadoop-xrootd connector in order to make the tool more automatically deployable, avoiding manual configuration. Both the developments are currently a work in progress.

5. Conclusions

We presented studies of executing the traditional HEP analysis workflow on Apache Spark. The efficiency of the application of the new tools has been evaluated both quantitatively, by measuring the performances, and qualitatively, focusing on the user experience. Our studies identified some bottlenecks and underlined the need to scale up the Spark infrastructure and to generalize the site configuration. The scaling behavior results are promising and they are only a factor of two from the original goal of reducing 1 PB to 5 hours, reasonably achievable with more hardware and further software optimizations.

6. Acknowledgments

We would like to thank the CMS collaboration and the LHC to provide the data for the use case and the ROOT based workflow. This work was partially supported by Fermilab operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy, and by the National Science Foundation under grant ACI-1450377 and Cooperative Agreement PHY-1120138.

References

[1] Zaharia M, Chowdhury M, Franklin M J, Shenker S and Stoica I 2010 *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* HotCloud'10 (Berkeley, CA, USA: USENIX Association) pp 10–10 URL http://dl.acm.org/citation.cfm?id=1863103.1863113

[2] Gutsche O, Cremonesi M, Elmer P, Jayatilaka B, Kowalkowski J, Pivarski J, Sehrish S, Surez C M, Svyatkovskiy A and Tran N 2017 *CoRR* **abs/1703.04171** URL http://arxiv.org/abs/1703.04171

[3] Elmer P, Hegner B and Sexton-Kennedy L 2010 *J. Phys. Conf. Ser.* **219** 032022

[4] Brun R and Rademakers F 1997 *Nuclear Instruments and Methods in Physics Research Section A* **389** 81 – 86 ISSN 0168-9002 new Computing Techniques in Physics Research V URL http://www.sciencedirect.com/science/article/pii/S016890029700048X

[5] Chatrchyan S e a (CMS Collaboration) 2008 *JINST* **3** S08004

[6] Evans L and Bryant P 2008 *Journal of Instrumentation* **3** S08001 URL http://stacks.iop.org/1748-0221/3/i=08/a=S08001

[7] Apache avro URL http://avro.apache.org

[8] Shvachko K, Kuang H, Radia S and Chansler R 2010 *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* MSST '10 (Washington, DC, USA: IEEE Computer Society) pp 1–10 ISBN 978-1-4244-7152-2 URL http://dx.doi.org/10.1109/MSST.2010.5496972
[9] Scala URL http://www.scala-lang.org
[10] Pivarski J, Svyatkovskiy A, Schenck F and Engels B 2016 histogrammar-python: 1.0.0 URL https://doi.org/10.5281/zenodo.61418
[11] Khristenko V and Pivarski J 2017 diana-hep/spark-root: v0.1.14_pre1 release URL https://doi.org/10.5281/zenodo.1019880
[12] Eos: Large disk storage at cern URL https://eos.web.cern.ch
[13] Motesnitsalis V hadoop-xrootd-connector URL https://gitlab.cern.ch/awg/hadoop-xrootd-connector
[14] Dorigo A, Elmer P, Furano F and Hanushevsky A 2005 Xrootd- a highly scalable architecture for data access WSEAS Transactions on Computers
[15] Apache parquet URL https://parquet.apache.org/
[16] Mascetti L, Labrador H G, Lamanna M, Mościcki J and Peters A 2015 *Journal of Physics: Conference Series* **664** 062037 URL http://stacks.iop.org/1742-6596/664/i=6/a=062037
[17] Cern openlab/intel cms big data project URL https://cms-big-data.github.io