Astronomical Data Analysis Software and Systems XXVI
ASP Conference Series, Vol. 521
Marco Molinaro, Keith Shortridge, and Fabio Pasian, eds.
© 2017 Astronomical Society of the Pacific

FACT-Tools – Processing High-Volume Telescope Data

Jens Buss,[1] Christian Bockermann,[2] Jan Adam,[1] Max Ahnen,[3] Dominik Baack,[1] Matteo Balbo,[4] Matthias Bergmann,[5] Adrian Biland,[3] Michael Blank,[5] Thomas Bretz,[3] Kai Bruegge,[1] Anton Dmytriiev,[4] Daniela Dorner,[5] Alexey Egorov,[2] Sabrina Einecke,[1] Christina Hempfling,[5] Dorothee Hildebrand,[3] Gareth Hughes,[3] Lena Linhoff,[1] Karl Mannheim,[5] Katharina Morik,[2] Sebastian A. Mueller,[3] Dominik Neise,[3] Andrii Neronov,[4] Maximilian Noethe,[1] Aleksander Paravac,[5] Felicitas Pauss,[3] Wolfgang Rhode,[1] Tim Ruhe,[1] Amit Shukla,[3] Fabian Temme,[1] Julia Thaele, [1] and Roland Walter[4]

[1] *Lehrstuhl für Experimentelle Physik 5, TU Dortmund;*
`jens.buss@tu-dortmund.de`

[2] *Lehrstuhl für künstliche Intelligenz, Informatik, TU Dortmund;*

[3] *ETH Zurich, Institute for Particle Physics;*

[4] *University of Geneva, ISDC Data Center for Astrophysics;*

[5] *Universität Würzburg, Institute for Theoretical Physics and Astrophysics*

Abstract. The amount of data produced in astroparticle physics exceeds the capacities of traditional data processing systems. Within the Big Data era, the field of computer science has brought up a plethora of tools and methods to scale data processing to a large number of nodes. In the stress field of rapid prototyping scientific algorithms and the implementation of scalable processing pipelines, we propose an abstraction for high-level modelling of such pipelines. This enables physicists to concentrate on their domain specific use-case at hand, while maintaining the benefit to deploy their algorithms within a scalable execution platform.

In this paper, we outline the collaborative work of physicists and computer scientists to develop a modeling framework that fits the aforementioned setting. The framework is used within the FACT project, showing its real-world practicability and ease-of-use.

1. Introduction

Several large experiments such as FACT, HESS, MAGIC, VERITAS, or the upcoming CTA project deploy high-speed cameras in Cherenkov telescopes to monitor astrophysical objects, such as AGNs, SNRs or GRBs. The First G-APD Cherenkov Telescope (FACT) is pioneering the use of solid state photo detectors for imaging atmospheric Cherenkov telescopes (IACT) and showed the reliability of silicon photo multipliers for earth-bound gamma-ray astronomy (Anderhub et al. 2013). Since October 2011, FACT is monitoring blazers at TeV energies in the northern sky and collected more then 7000 hours of data.

The amount of data collected by modern IACTs poses big challenges for the data storage and the data analysis. The challenges range from domain specific physics aspects, such as finding good filtering algorithms/parameters for background rejection, to scalability issues, requiring analytical software to be scaled to large clusters of compute nodes for an effective real-time analysis. Modern cluster environments, which emerged from the Big Data community, aim at distributed data storage with a strong emphasis on *data locality* and *fault-tolerant* computing. These clusters perfectly match the requirements of modern data-driven physics experiments. However, their programming demands expert knowledge to gain the full performance advantages at the user level. In a joint effort of physicists and computer scientists we targeted this area of conflict using the generic *streams* framework, a plug-able data processing environment developed at the Collaborative Research Center SFB-876. Using *streams* allows for the high-level design of analytical data flows, while maintaining compatibility to large scale streaming platforms. This enables physicists to develop and test new algorithms in a local environment and deploy these on modern compute clusters without adaptions.

In the following, we will outline the basic principles behind a Big Data approach taken for the FACT experiment. In Section 2 we describe the basic data processing steps and the high-level abstraction made within the *streams* framework. Following that we elaborate the re-use of analytical blocks in different Big Data platforms in Section 3 and conclude the paper.

2. High-Volume Data Processing Pipeline

The data analysis process of many astroparticle experiments can naturally be described by the processing steps applied to the real data flow. Figure 1 shows the pipeline of the FACT experiment at a high abstraction level. Despite the data flow denoted by the green arrows, there are substantial dependencies within the different task, that require careful interaction while developing algorithms and functions for each block of the pipeline. As an example, the *cleaning* step in the beginning selects the shower pixels within a recorded event. This selection is performed by a heuristic algorithm and its output affects the following steps, i.e. the *feature extraction*. The development and implementation of a software system for this pipeline therefore requires a lot of fine tuning of each step, ultimately affecting other parts of the pipeline. This requires extensive background knowledge and is usually a domain expert task, i.e. algorithm development by physicists.

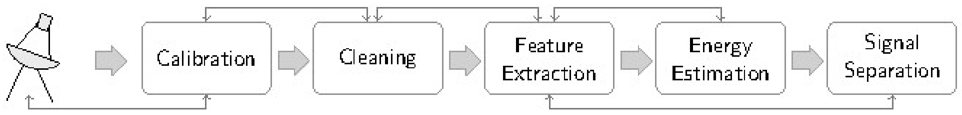

Figure 1. The data processing steps within the FACT experiment.

As another important aspect, the system needs to be able to handle high loads of data to match the volume of events provided by the telescope. Modern Big Data system employ a variety of distributed and data-parallel approaches to scale processing pipelines along with the availability of additional computing nodes. The scaling to distributed compute clusters is a trending area of research in computer science.

To combine this two distinct areas – the development of appropriate algorithms for the physics problems and the high-performance execution on large clusters – we use the *streams* framework as proposed in Bockermann (2015). The idea of *streams* is to decouple the software development part from the execution of the pipeline. By the use of a simple, generic API, the physicists are enabled to formulate their algorithms as modular streaming functions. Usually, each of these functions is provided by a single Java class. Using a declarative XML modelling approach allows for specifying a data flow graph that reflects a pipeline based on these modular functions.

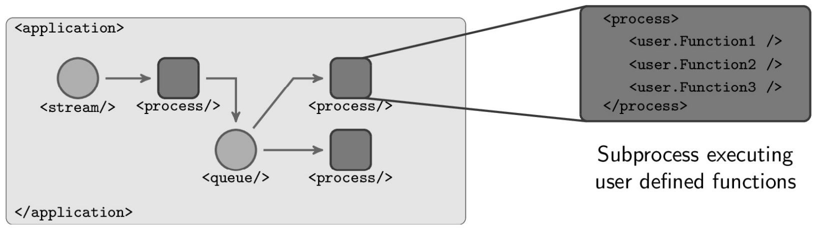

Figure 2. The outline of an application in streams – a graph of connected processes.

3. FACT on Big Data Platforms

In the context of Big Data, several platforms for massive parallel processing have been proposed. Most popular is the Map-Reduce revival within the *Apache Hadoop* (Cutting et al. (2007)) framework and its dependent *Apache Spark* (Zaharia et al. (2010)). It features parallel batch processing of huge data volumes stored in its distributed file system. The need for real-time processing of data fostered the development of streaming platforms, most prominent being the Apache Storm system (Marz et al. (2011)). The batch as well as the streaming approach both complement each other, which has led to the so-called *Lambda Architecture* proposed in Marz & Warren (2014). This architecture features three different layers: the *speed layer* for real-time computation, the *batch layer* for processing of archived data and the *service layer* as a representation of results obtained from the other two layers.

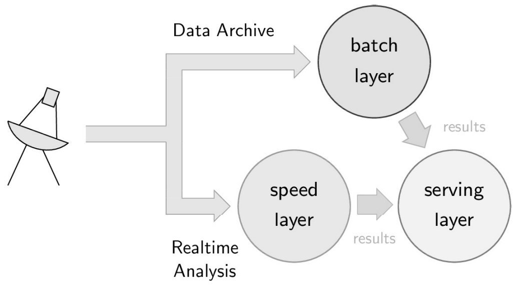

Figure 3. The *Lambda Architecture* in the context of telescope data.

As can be seen in Figure 3, the data recorded by the telescope is being analyzed in real-time as well as archived to a batch system. The real-time analysis is usually performing a lower precision analysis whereas the batch processing of archived data allows for a more in-depth analysis of the data.

There exists several additional implementations and platforms for each of the layers. The key objective of building a tool chain based on the *streams* framework – as has been developed with the *FACT Tools* – is to abstract from the underlying engine and allow for user functions to be mapped to different platforms without changes in the code. Providing an embedding of the *streams* runtime environment for systems like Apache Storm or Apache Hadoop/Spark, allows for the design of data flows using XML and a flexible re-use of the existing code library as shown in Figure 4.

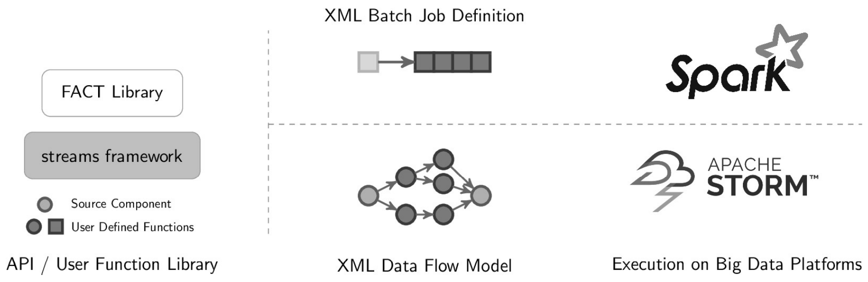

Figure 4. XML Modelling for batch and stream processing with existing components.

4. Summary & Conclusion

Recently, the *Lambda Architecture* has become a state-of-the-art model for large scale data analysis. Based on the *streams* framework, we successfully implemented a modelling tool for data stream processing (i.e. the speed layer) and applied it to the FACT telescope (Bockermann et al. (2015)). The use of XML in *streams* provides a flexible way for rapid-prototyping of new pipelines, testing out different pre-processing steps and sharing the overall processes with the involved scientists.

By developing an additional embedding of the *streams* framework for the Apache Spark system, we were able to re-use all of the existing domain specific code to run batch jobs on large amounts of archived data on a large-scale distributed system powered by Apache Spark, effectively minimizing the overall processing time.

Acknowledgments. This work has been supported by the DFG, Collaborative Research Center SFB 876, project C3 (`http://sfb876.tu-dortmund.de/`).

References

Anderhub, H., et al. 2013, JINST, 8, P06008
Bockermann, C. 2015, Ph.D. thesis, TU Dortmund University. URL `https://eldorado.tu-dortmund.de/handle/2003/34363`
Bockermann, C., et al. 2015, in Proceedings of the European Conference on Machine Learning (ECML) (Springer Berlin Heidelberg)
Cutting, D., et al. 2007, Apache Hadoop. `http://hadoop.apache.org/`
Marz, N., & Warren, J. 2014, Big Data - Principles and best practices of scalable realtime data systems (Manning Publications Co.)
Marz, N., et al. 2011, Apache storm. `https://storm.apache.org/`
Zaharia, M., et al. 2010, in Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, 10