Distributed Real-Time Data Stream Analysis for CTA

Kai Brügge[1], Alexey Egorov[1], Christian Bockermann[1], Katharina Morik[1] and Wolfgang Rhode[1]

[1]*TU Dortmund, Dortmund, Germany;*
`kai.bruegge@tu-dortmund.de`
`alexey.egorov@tu-dortmund.de`

Abstract.
Once completed, the Cherenkov Telescope Array (CTA) will be able to map the gamma-ray sky in a wide energy range from several tens of GeV to some hundreds of TeV and will be more sensitive than previous experiments by an order of magnitude. It opens up the opportunity to observe transient phenomena like gamma-ray bursts (GRBs) and flaring active galactic nuclei (AGN). In order to successfully trigger multi-wavelength observations of transients, CTA has to be able to alert other observatories as quickly as possible. Multi-wavelength observations are essential for gaining insights into the processes occurring within these sources of such high energy radiation.

CTA will consist of approximately 100 telescopes of different sizes and designs. Images are streamed from all the telescopes into a central computing facility on site. During observation CTA will produce a stream of up to 20 000 images per second. Noise suppression and feature extraction algorithms are applied to each image in the stream as well as previously trained machine learning models. Restricted computing power of a single machine and the limits of network's data transfer rates become a bottleneck for stream processing systems in a traditional single-machine setting. We explore several different distributed streaming technologies from the Apache Big-Data eco-system like Spark, Flink, Storm to handle the large amount of data coming from the telescopes. To share a single code base while executing on different streaming engines we employ abstraction layers such as the streams-framework. These use a high level language to build up processing pipelines that can transformed into the native pipelines of the different platforms. Here we present results of our investigation and show a first prototype capable of analyzing CTA data in real-time.

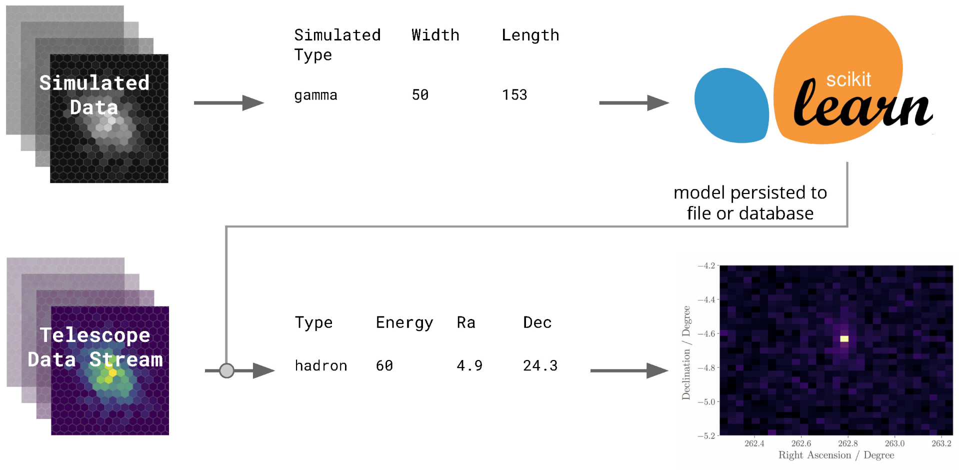
1. The Cherenkov Telecope Array

Once completed, the Cherenkov Telescope Array (CTA) will be able to map the gamma-ray sky in a wide energy range from several tens of GeV to some hundreds of TeV and will be more sensitive than previous experiments by an order of magnitude. CTA will consist of approximately 100 imaging air cherenkov telescopes (IACTs) of three different sizes. Telescope data will be streamed via network from the telescopes to a central computing facility on-site. Cherenkov telescopes record light produced by particle showers induced in the upper atmosphere by cosmic rays. While the cosmic ray flux is approximately isotropic over the sky, gamma rays of cosmic origin can be pinpointed back to its source. Filtering air showers produced by cosmic rays while keeping those produced by gamma rays is the big challenge in IACT data analysis.

2. Real Time Analysis

One of CTA's main goals is monitoring the sky for transient events. These include Gamma-Ray Burst (GRBs) events and Active Galactic Nuclei (AGNs). To gain more understanding about the physics working in GRBs and AGN it is vital to perform multi-wavelength observations. In case a GRB is detected, CTA can alert other experiments to trigger observations in other wavelength bands. The data aquisition system will supply the real time analysis (RTA) with calibrated images of each triggered telescope. Features for classication/regression are calculated on each image in the triggered events. Simulated and labeled data is used to train the models using the Python machine learning library `scikit-learn` (Pedregosa et al. 2011). These models are then used for filtering of comsic ray showers and estimation of primary particle energy. The trained `scikit-learn` models are converted into the PMML (Guazzelli et al. 2009) format using the `sklearn2pmml` (Ruusmann 2015) library. This way the stored model can be shared between programming languages and applied to the data stream from the telescopes.

The expected event rate of CTA will be between 10000 and 20000 events per second depending on deployment and pointing direction (Paz Arribas et al. 2012). Every real time processing system for CTA data will have to handle that data rate.



Simulated Type	Width	Length
gamma	50	153

Type	Energy	Ra	Dec
hadron	60	4.9	24.3

Figure 1. Overview of the analysis process. We use `scikit-learn` to train models on simulated data and apply these models to the data stream coming from the telescopes to get results in near real time.

3. Machine Learning Performance

Supervised machine learning is used to separate signal from background and to estimate the energy of a recorded event. Features for classification/regression are calculated from raw input images. We trained a Random-Forest (Breiman 2001) classifier with 200 trees on simulated data to separate signal events from background events.

The trained model is then applied to each image from the telescopes in the data stream. Figure 2 presents the results of the model application. The plot on the left shows the performance of background suppression for the three different telescope types. The predictions for each telescope are then averaged to get a combined prediction for the entire event. This improves background rejection significantly as shown in the right image.

Energy estimation is performed in a similar manner. A Random Regression Forest is trained on the same simulated data. The resulting model is then applied to the data stream and predictions are averaged for all images in a single event.

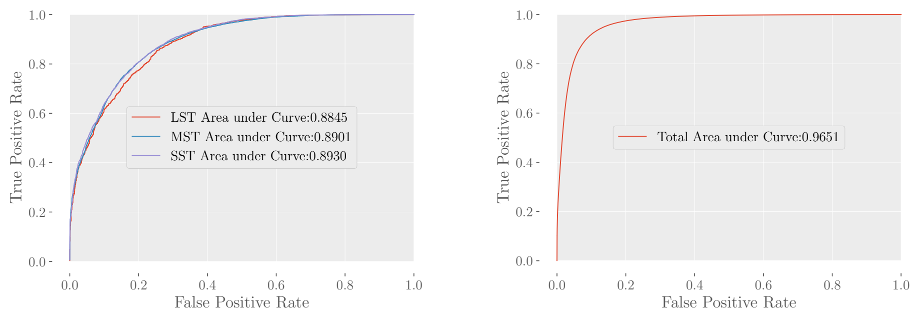


Figure 2. *Left:* Performance of background suppression for the different telescope types. The small size telescope (SST), the mid size telescope (MST) and the large size telescope (LST). *Right:* Improved background suppression of a combined predictor when averaging the predictions for all telescopes in a CTA event.

4. Runtime Performance

Frameworks for distributed stream processing such as Apache Storm (Toshniwal et al. 2014) or Apache Flink (Alexandrov et al. 2014) allow for workload distribution with fault tolerance and high availability mechanisms to recover from hardware or network failures. Tuning Apache Spark for fast streaming applications is more complicated due to its micro-batch architecture. We compare the runtime of Storm and Flink on a single core to measure the impact of the overhead these frameworks produce. To share a single code base while executing on different streaming engines we employ the `streams-framework` (Bockermann 2015) as an abstraction layer. This way the analysis pipeline is only defined once, but can be easily executed on top of different distributed streaming engines. The left image in Figure 3 shows that Flink shows less runtime overhead compared to Storm.

We continue our experiments on Flink due to faster processing, easier setup and a more comfortable high-level API compared to Storm. Any real-time processing system for CTA has to be able to handle the data rate of up to 20000 events per second and provide analysis results within 30 seconds according to the official CTA requirements.

The right Figure 3 presents the evaluation of a full CTA pipeline executed on top of Flink. For this test a machine with 24 physical CPU cores was used. The datarate goes up to approximately 14000 events per second on this single machine. We use a simple self hosted Flink cluster on two machines with 24 cores each to scale the process up. This way event rates of more than 20000 events per second are achieved.

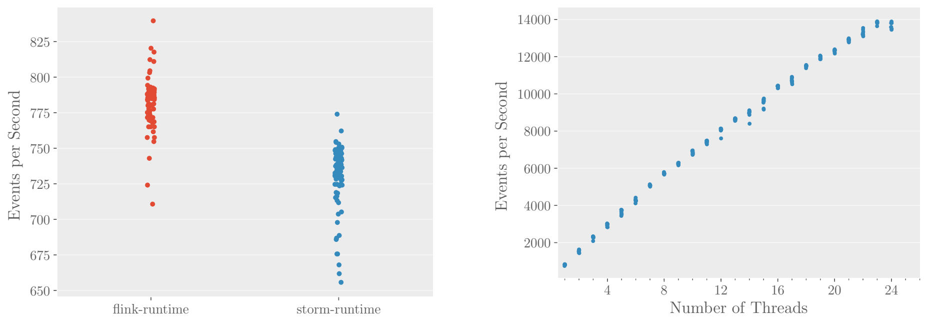


Figure 3. *Left:* Overhead produced by distributed processing frameworks. *Right:* Throughput of CTA analysis pipeline executed on multiple cores.

Acknowledgments. Part of this work is supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project C3.

References

Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J.-C., Hueske, F., Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., Naumann, F., Peters, M., Rheinländer, A., Sax, M. J., Schelter, S., Höger, M., Tzoumas, K., & Warneke, D. 2014, The VLDB Journal, 23, 939

Bockermann, C. 2015, Ph.D. thesis, TU Dortmund University. URL `https://eldorado.tu-dortmund.de/handle/2003/34363`

Breiman, L. 2001, Machine learning, 45, 5

Guazzelli, A., Zeller, M., Lin, W.-C., Williams, G., et al. 2009, The R Journal

Paz Arribas, M., Schwanke, U., Wischnewski, R., & CTA Consortium, f. t. 2012, ArXiv e-prints. `1211.3061`

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. 2011, Journal of Machine Learning Research, 12, 2825

Ruusmann, V. 2015, Python library for converting scikit-learn models to pmml. URL `https://github.com/jpmml/sklearn2pmml`

Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., & Ryaboy, D. 2014, in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (New York, NY, USA: ACM), SIGMOD '14, 147. URL `http://doi.acm.org/10.1145/2588555.2595641`