HTC for Gamma-Ray Astronomy on Low-Power Platforms

A. Madonna[1,2], D. Bastieri[2,3], M. Mastropietro[1], L.A. Antonelli[1], S. Lombardi[1], S. Amerio[3] & D. Lucchesi[3]

1. *Rome Observatory, INAF, Rome, Italy*
2. *CUDA Research Center, University of Padova, Italy*
3. *Dept. Physics and Astronomy, Univ. Padova and INFN, Padova, Italy*

Abstract

Detectors for Gamma-ray Astronomy are the prototypes for distributed experiments. Single detectors may be scattered in an area of few square kilometres, and the capability of each unit to process, at least partially, its own data before sending them to the central data acquisition provides a key advantage. We aim at developing and testing algorithms and techniques to implement such kind of local data sparsification at detector level. To reach this goal, we leveraged the parallel capabilities of Kayla.

(Gamma-ray) Astronomy and Low-Power Computing

Revolution is a term borrowed from Astronomy. Originally used by the Polish mathematician and astronomer Nicolaus Copernicus (1473 – 1543) to describe that it was the Sun rather than the Earth at the center of the Universe[1], it soon entered everyday language to mean a drastic and far-reaching change of paradigm. Astronomy itself underwent a deep revolution when modern computing allowed new, powerful algorithms to be executed within reasonable execution times. The new frontier is indeed Low-Power Computing, as future experiments like CTA, the Cherenkov Telescope Array[2], will collect data at rate of hundreds of GB/s.

CTA, and, as a matter of facts, any other conceivable astronomical observatory, has to operate in a secluded or desert location to avoid light or, more generally, electromagnetic *pollution*. It is apparent that analyzing data virtually in *real-time* would be a nice feature, mainly because flaring or anomalous behaviors of celestial bodies could be detected sooner, allowing to broadcast notices about the events to a wider astronomical community that could readily interact and observe the object with different instruments before it fades. Transmitting data at almost 1 TB/s toward a data center that is able to handle the data in real-time is soon ruled out because of the cost, and the only viable solution we are left with is to analyze data *on-site*, shifting the cost burden from data transfer to power consumption.

The algorithms involved are usually quite simple, but have to be applied to many independent pixels, effectively entering the realm of High-Throughput Computing (HTC). In the following, we will discuss the opportunity offered by Kayla and successors to perform HTC at low-power.

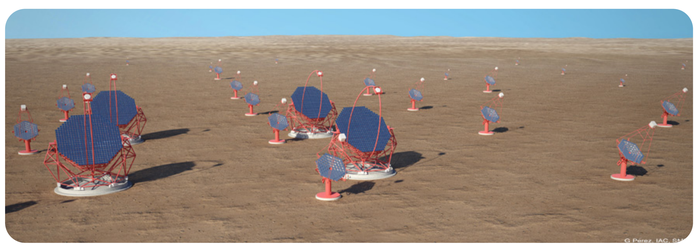


Figure 1. Artist's impression of the Cherenkov Telescope Array (CTA), which will be built at two sites – one in the northern hemisphere and one in the southern hemisphere. It will include telescopes of different sizes to capture the Cherenkov light from showers produced by gamma rays across an extended energy range. (Image credit: G. Pérez/IAC/SMM.)

[1] His model was published in the book "*De revolutionibus orbium coelestium*" (On the Revolutions of the Celestial Spheres), Johannes Petreius, Nuremberg 1543.
[2] For more information about the experiment, see http://www.cta-observatory.org

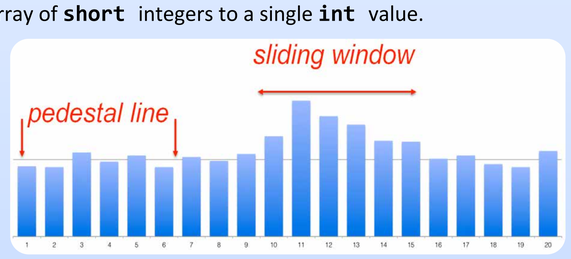
Data Crunching: the recipe

An average telescope of the CTA matrix could work at a rate of ~200 Hz. Data are transferred onto a disk in chunks of 2 GB. At 200 Hz a single telescope will need around 50 s to collect 2 GB of data (S. Lombardi: tests on MAGIC II telescope). Therefore, to be in *real-time* we should be able to reduce the data chunk in <50 s. But we want to do more than that: We want to crunch the data in *real-time* using as low energy as possible.

Procedure

We start with a preliminary evaluation of the pedestal offsets from 200 random events; since this is a one-time process on a definite amount of data, it has no impact on the overall timing of the on-line data crunching.

At this point, we start with the actual analysis chain: the data flow in from the telescope, we subtract the pedestal and we integrate the signal in a sliding window: an interval of fixed width sweeps through the data of a given event, gradually integrating its samples. As a result of this step, data is converted from an array of `short` integers to a single `int` value.



Signal calibration is then carried out, transforming the integer ADC counts data into a floating point Photoelectron Equivalent (phe) data. In this phase it is crucial to exploit the Fused Multiply–Add (FMA) operation, performing a multiplication and an addition in a single step, with a single rounding (e.g. `$0 = $0×$1+$2`). Since many modern processors provide an hardware implementation for FMA, there is virtually no difference in timing between pedestal subtraction and subtraction+calibration.

Photoelectron equivalents are then sorted, clustered and cleaned, removing any non-relevant observation, and the first 10 momenta are evaluated.

The process is completed and the data chunk is output back to the storage device.

Kayla test system

Development board by SE.CO.
CPU: NVIDIA Tegra3 ARM Cortex-A9 SoC
GPU: NVIDIA Quadro K2000

Power consumption:
Tegra3: ≈ 5W
GPU: 12V rail @ ≤ 1.6A ⇒ $\langle P \rangle \leq 20W$

Data Reduction on Kayla

We equipped the Kayla board with a solid state disk (128 GB by Plextor). Preliminary tests showed that the transfer between the SSD and the Kayla board can be easily sustained at ~1 Gb/s (or ~15 s for the 2 GB chunk), leaving us with something like 35 s to reduce the data. It must be noted that output does not really need much time, being the output 3 to 4 orders of magnitude smaller than the input.

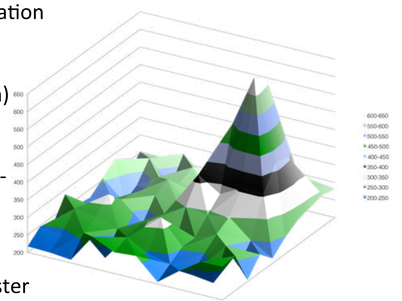
Mainly because of the FMA, the Tegra3 CPU takes 25-30 s to complete pedestal subtraction, the signal integration and calibration.

GOOD NEWS: **Everything done at 5W!**
BAD NEWS: **End of the time budget!**

Introducing clustering on K2000:

The transfer between CPU and GPU dominates over other processing phases, however, our 30 s budget can also accommodate for:

1) Pedestal subtraction + calibration
2) Pixel sorting
3) Set **HI** threshold (fixed or at a given % of the maximum)
   ⇒ pixels above the **HI** threshold are called *seeds.*
4) Check if pixels that are neighbors of *seeds* are above a **LO** threshold
   ⇒ those pixels will be grouped together in a cluster
   ⇒ pixels not belonging to any clusters are set to zero
5) Evaluate first 10 image momenta



With a data throughput of 2 GB every 50 s, the ARM processor has just enough time to subtract the pedestals and calibrate the data (transform the time slices expressed in ADC counts into a single photoelectron equivalent: item # 1), with a power budget of 5W. Clustering/cleaning (items #2..4) and computing momenta (item #5) within the given time interval is feasible only if run on the GPU, adding about 20 W (peak) to the total power consumption. ***The issue is now that if we resort to the GPU, much of the computing power at our disposal remains unused!*** Measurements on the 12 V rail indicates that the K2000 absorbs power for about 15 s, in line with the measurement of the average power: 12 W. It is our opinion that speeding up the transfer rate, for instance implementing a PCIe v3 instead of the Kayla v1, would still not make the board able to perform the whole analysis at 5 W within the given time budget.

Conclusions

Our tests demonstrate that pedestal processing and calibration with a sustained data flow of 1 Gb/s are feasible on an ARM CPU (in this case the Tegra3) consuming just 5 W. Exploiting the easy expandability of the Kayla system, additional analyses could still be done with just a small power increase using for example a specialized co-processing hardware, like the NVIDIA K2000. We look forward to testing the analysis chain on the Jetson TK1 and/or similar architecture of the Maxwell and following series.