Low Power Computing in Gamma-Ray Astronomy

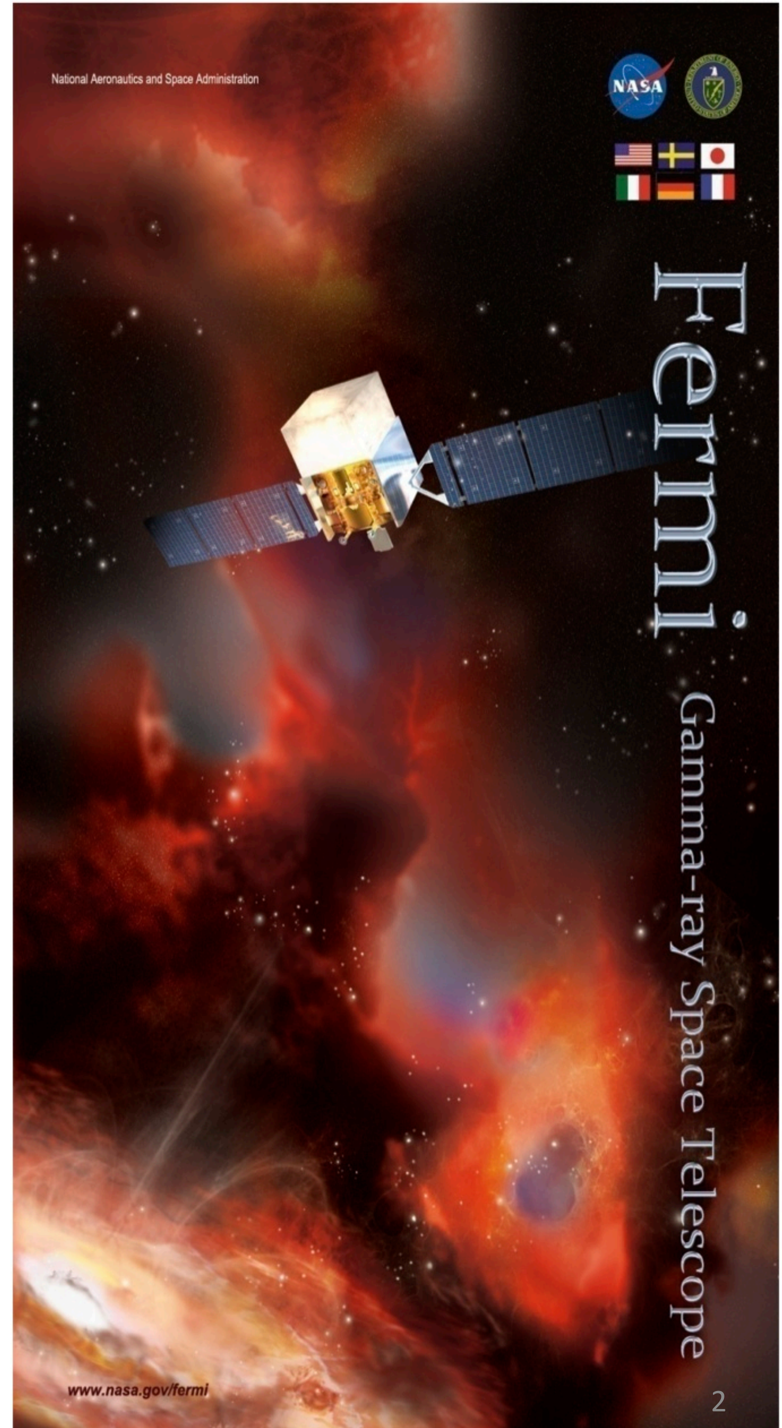
Denis Bastieri

GPU Research Center – University of Padova
INFN Padova

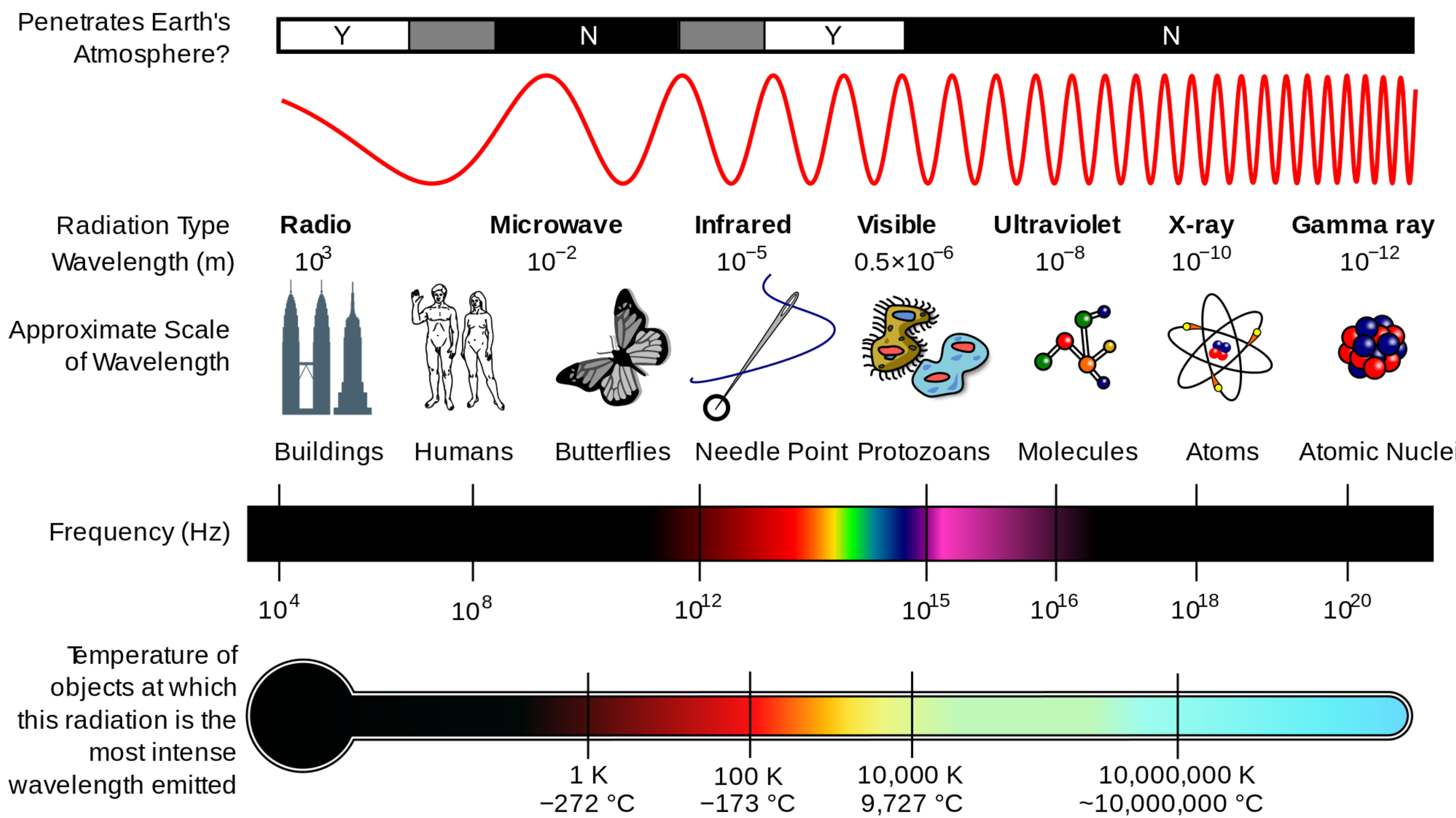
CoLA Workshop – Ferrara – February 25th, 2016

Outline

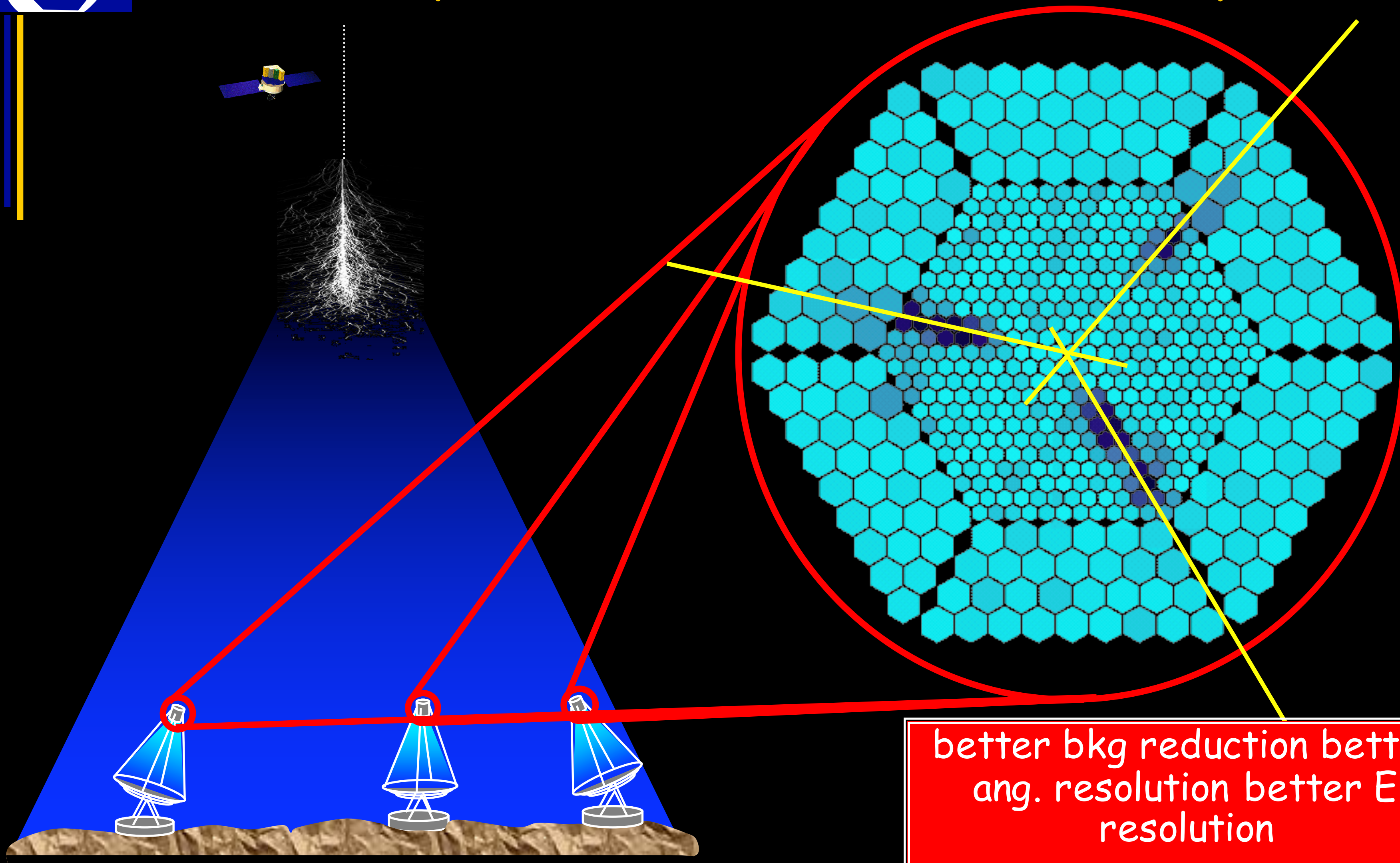
- Fermi for *Fermi*
  - Analysis chain
  - Exposure cube
  - Maximum likelihood
  - What’s next?
- LPC for HTC
  - Data crunching with ARM
  - Adding GPUs
  - What about FPGAs?
  - What’s next?



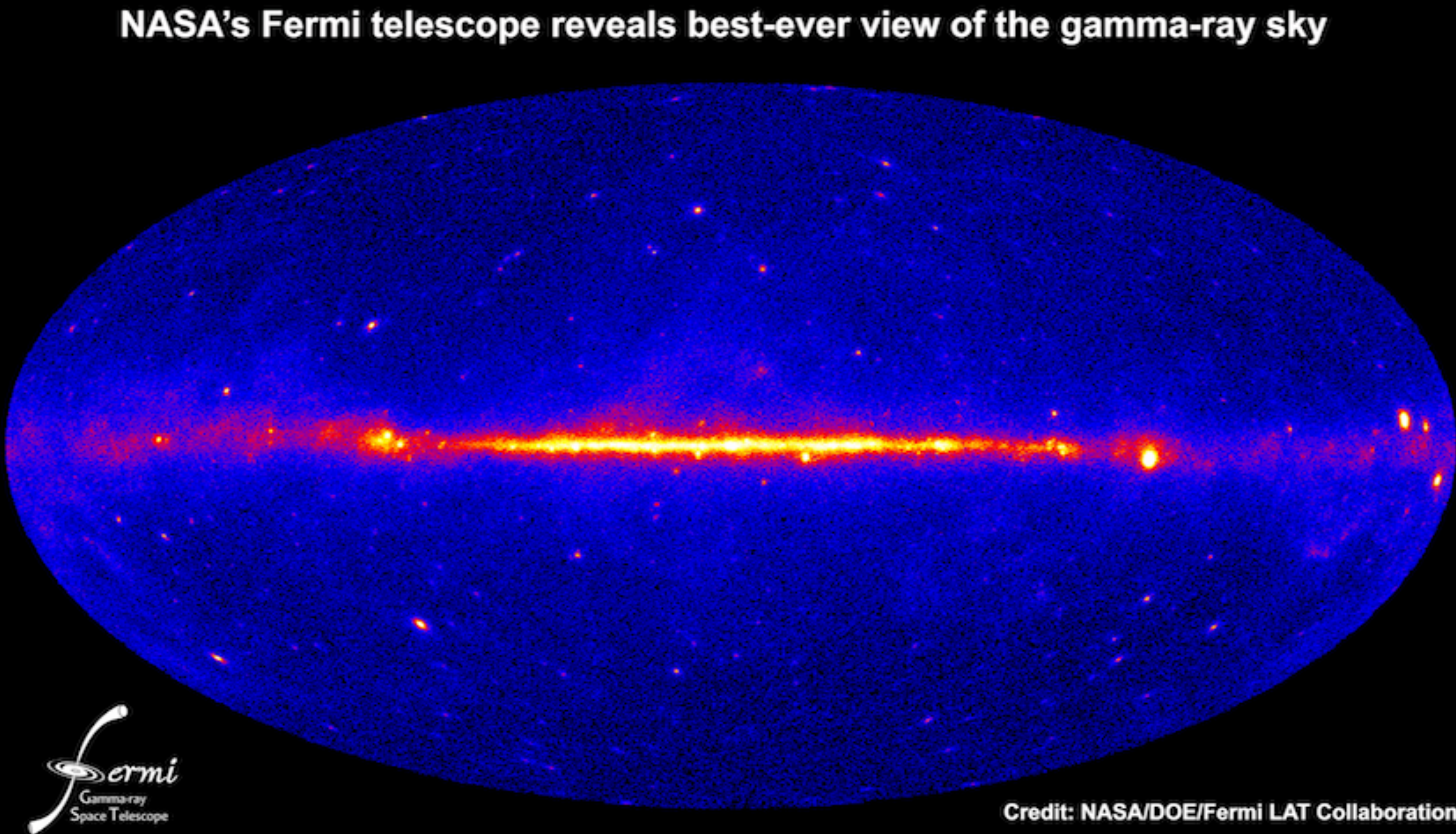
Electromagnetic spectrum



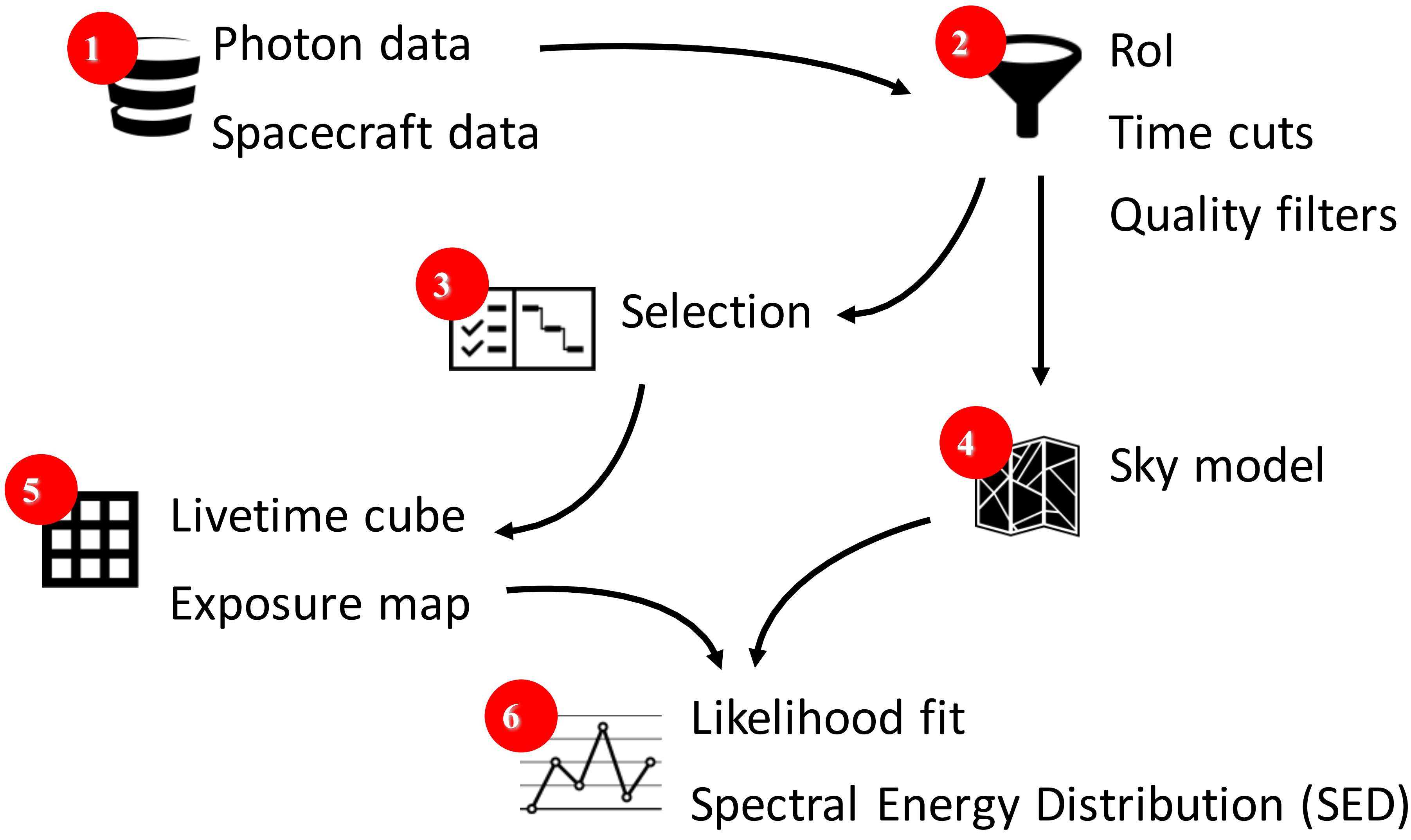
IACT – System of Cherenkov Telescopes



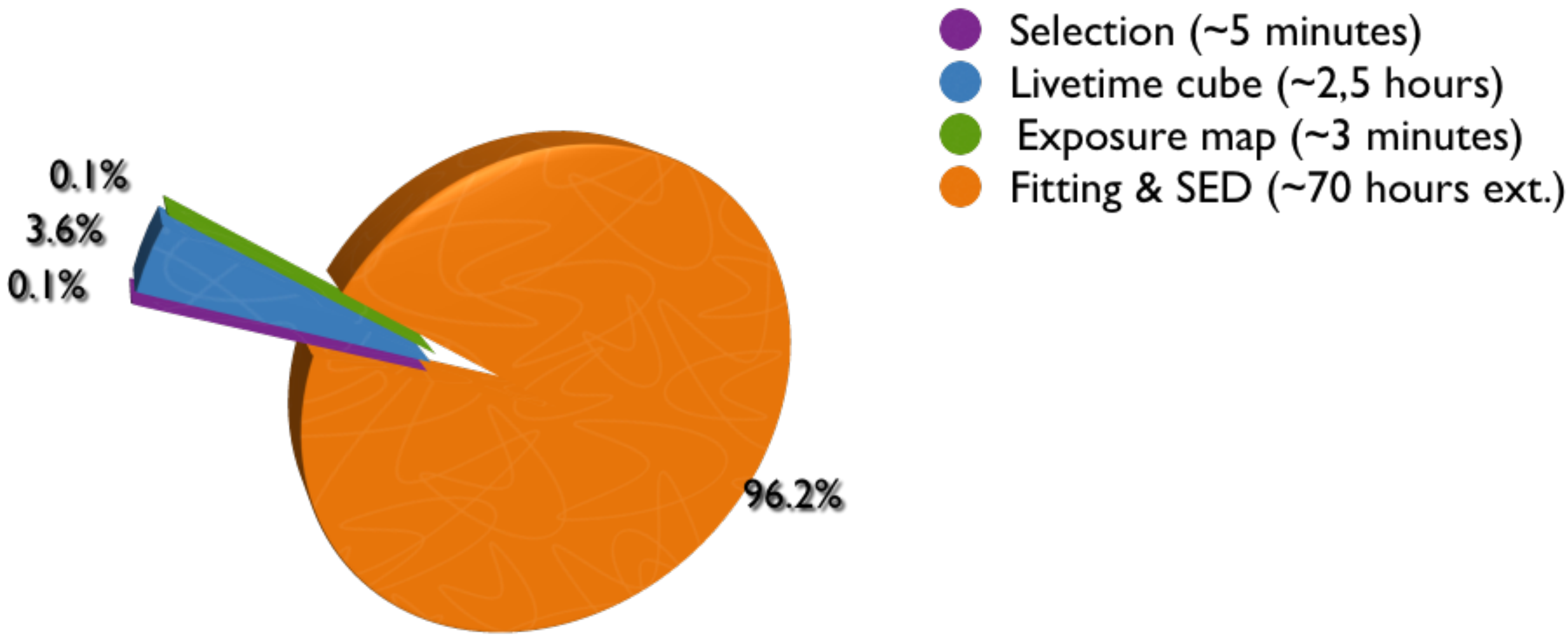
better bkg reduction better ang. resolution better E resolution


NASA's Fermi telescope reveals best-ever view of the gamma-ray sky
Fermi
Gamma-ray
Space Telescope
Credit: NASA/DOE/Fermi LAT Collaboration

Fermi LAT spectral analysis pipeline



Computational Cost

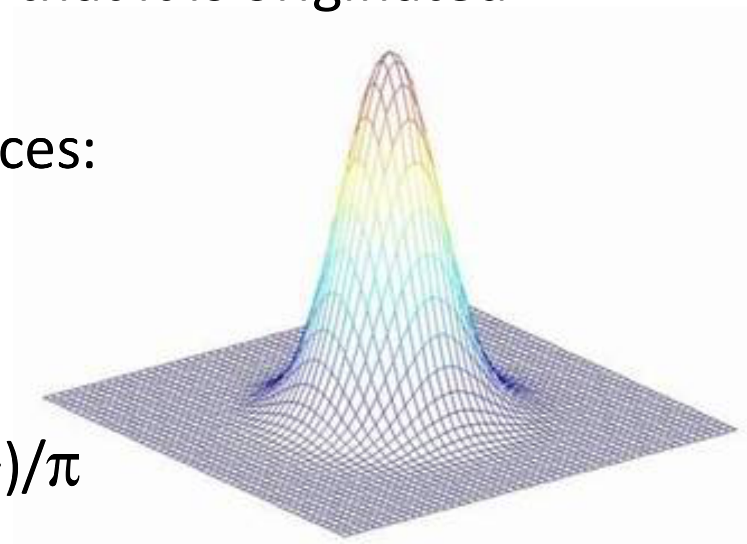


3 days per source for ~5 years worth of data

ScienceTools-09-31-00 · 2 × Xeon E5620 · 24 GB RAM

Basic recipe

- For every detected photons
  - Compute the probability that it is originated from the model
  - Easy to do for point sources:
    PSF ~ 2D-Gauß
    $\delta = \text{dist}[(RA_\gamma, dec_\gamma), (RA_{src}, dec_{src})]$
    $prob(RA_\gamma, dec_\gamma) \sim \exp(-\delta^2)/\pi$
  - Multiply all *p* together
    (actually *sum*: better using (−)log)
- Typically: $1M\gamma$/year

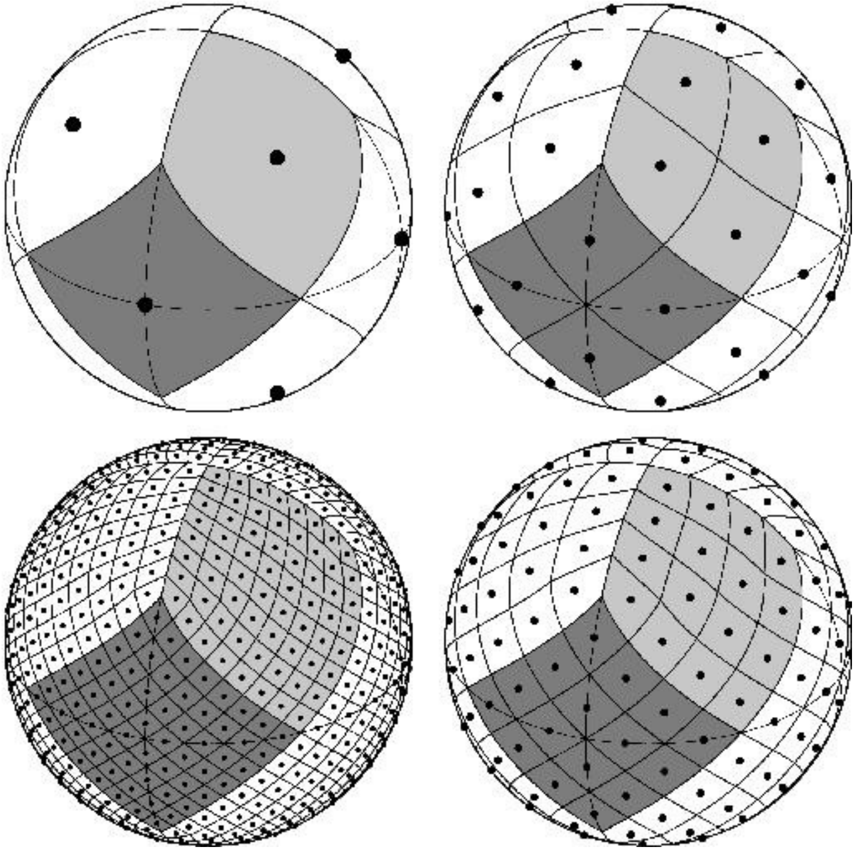


A new spectral analysis pipeline

GPU Livetime Cube tool

number of seconds under which a given direction is observed under a given angle

- HEALPix maps
- order 64 → 49152 pixels
- 40 bins of inclination
- **one thread per pixel**



Summer 2011 at SLAC NATIONAL ACCELERATOR LABORATORY

The kernels

- FT2 entries: [start, stop], ra_z, dec_z, lt, wlt
- For every GTI (10k)
  - GTI couples *cached* in shared
  - flag FT2 in GTI

```
kGTI<<<g_currentDbSize/512+1, 512>>>
```

- For every lt-bin (HPix, cos(θ))
  - Compute the angle between HPix and FT2 entry (entirely loaded into GPU memory).
  - Update the proper cos(q):
    - `cosbin = ceil(40E-6*rintf(1E6*sqrt(1-dot)));`
    - `ltcube += lt; wltcube += wlt;`

```
kEval2<<<96, 512>>
```

OAR, 30 Gennaio 2012

Denis Bastieri: astro
Andrea Pigato: web/daemon
Giorgio Urso: CUDA

Ultrafast
Robotic Interface
for Extended Likelihood

Urania CUDA Server in Padua

2 × S2050 cards

- 4 GPUs each, Fermi arch.
- 3 GB GDDR5 per GPU
- 448 CUDA cores per GPU

Likelihood: Algorithm details

Generate a set of parameters for each source, getting nearer to the minimum of *-log(Likelihood)* (~10^2 iterations)

→ For each observed photon ($10^4 \div 10^6$)

→ For each source of the sky model (~10^2)

→ Probability of the photon given source spectrum and parameters

→ Sum all *log*(probabilities) to obtain the new *Likelihood*

Likelihood: Algorithm details

Once *-log(Likelihood)* has been minimized

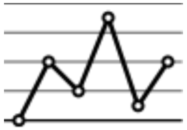
↳ Remove one source at a time from the model

  ↳ Refit

  ↳ Use Wilks's theorem to choose the right hypothesis

$$\mathrm{TS} = -2\ \log\frac{\mathscr{L}_0}{\mathscr{L}} = 2\left(\log\mathscr{L} - \log\mathscr{L}_0\right)$$

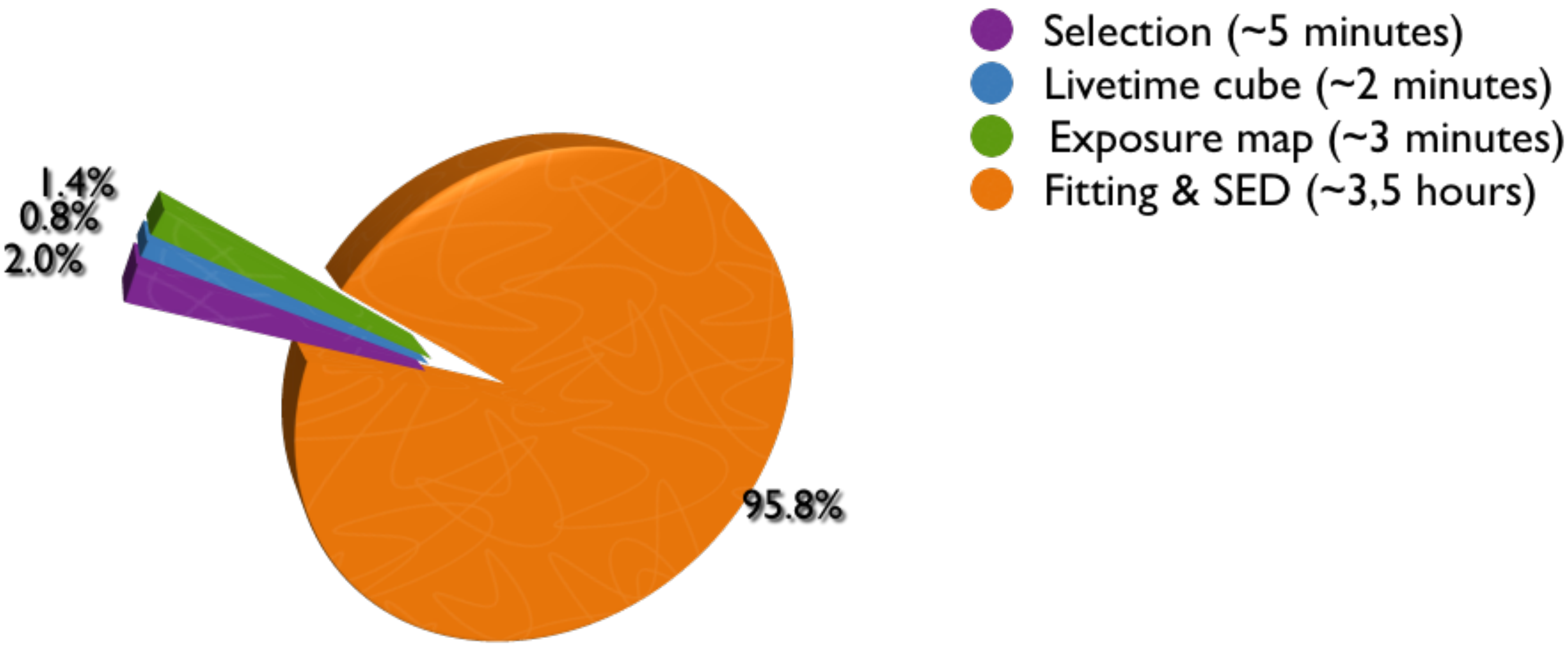
A new spectral analysis pipeline



GPU Likelihood fitting tool

- data cleaning
- data transferring (Host/GPU)
- aggressive caching
- **one thread per photon**
- modular system of template functions for spectra
- likelihood computation
- minimization by CERN's Minuit

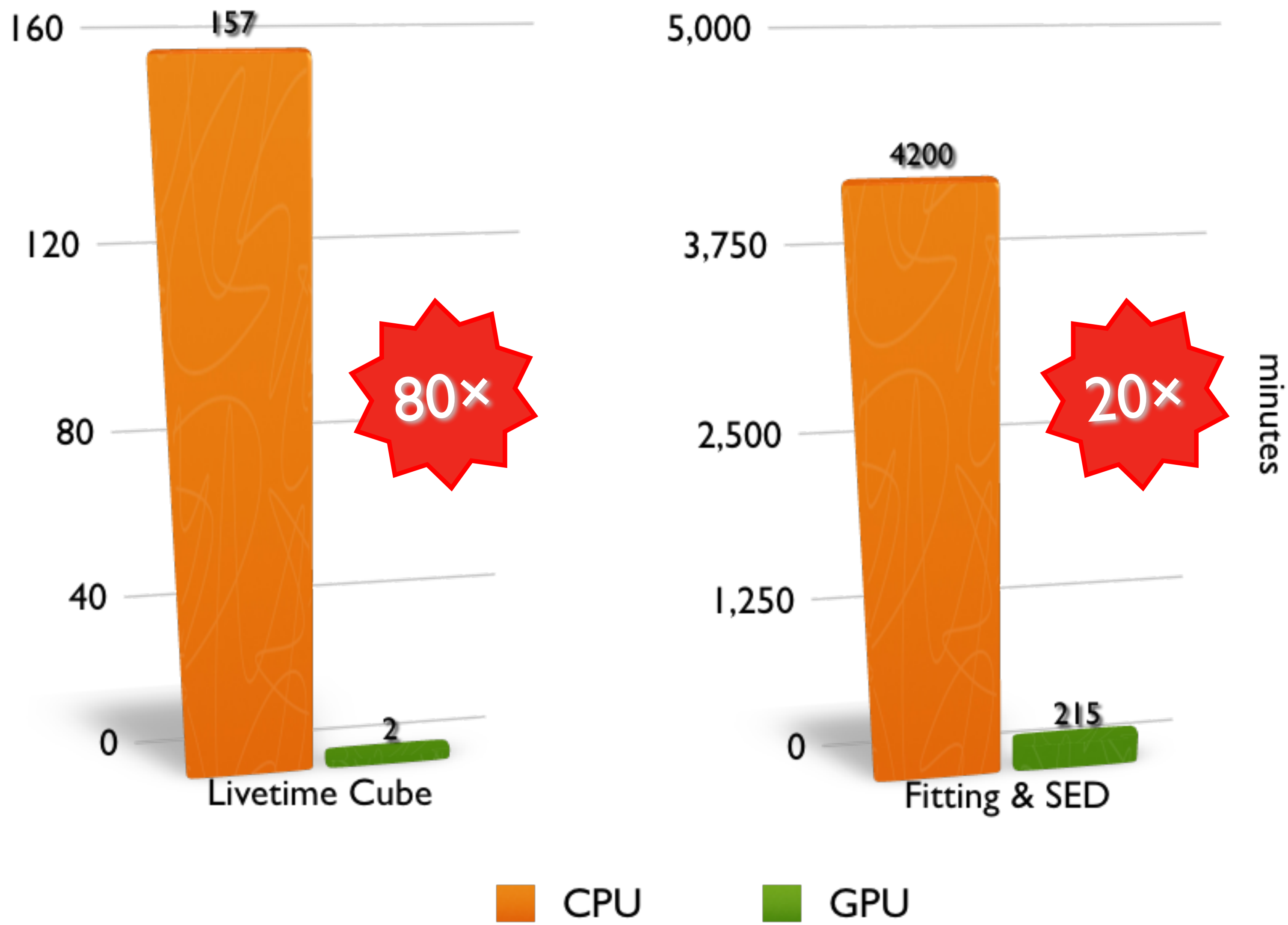
Computational Cost



from 3 days to **4 hours** per source for ~5 years worth of data

New Pipeline · NVIDIA S2050 · 3 GB RAM

Performance comparison



Maximum Likelihood Estimation on GPUs: Leveraging Dynamic Parallelism

M. Mastropietro[1], D. Bastieri[2,3], A. Pigato[2], A. Madonna[1,2], S. Amerio[3], D. Lucchesi[3], L.A. Antonelli[1] & G. Lamanna[4]

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*
4. *LAPP, Laboratoire d'Annecy-le-Vieux de physique des particules, Annecy, France*

Maximum Likelihood Approach

- Parameters estimation through maximization
- Hypotheses testing through Wilks's theorem

Null hypothesis max likelihood, h parameters

$$TS = -2 \log \frac{\mathscr{L}_0}{\mathscr{L}} \underset{N\to\infty}{\rightarrow} \chi^2_{m-h}$$

non fixed parameters

Alternative hypothesis max likelihood, m parameters

Unbinned Likelihood

Poisson statistics: $p(n,\lambda) = \frac{\lambda^n e^{-\lambda}}{n!}$

Total number of predicted photons

$$\log \mathscr{L}(\{\alpha_k\}) = \sum_{i\in P} \log J(E, \vec{p}; \{\alpha_k\}) - \Lambda_{tot}(\{\alpha_k\})$$

Set of bins with an observed photon

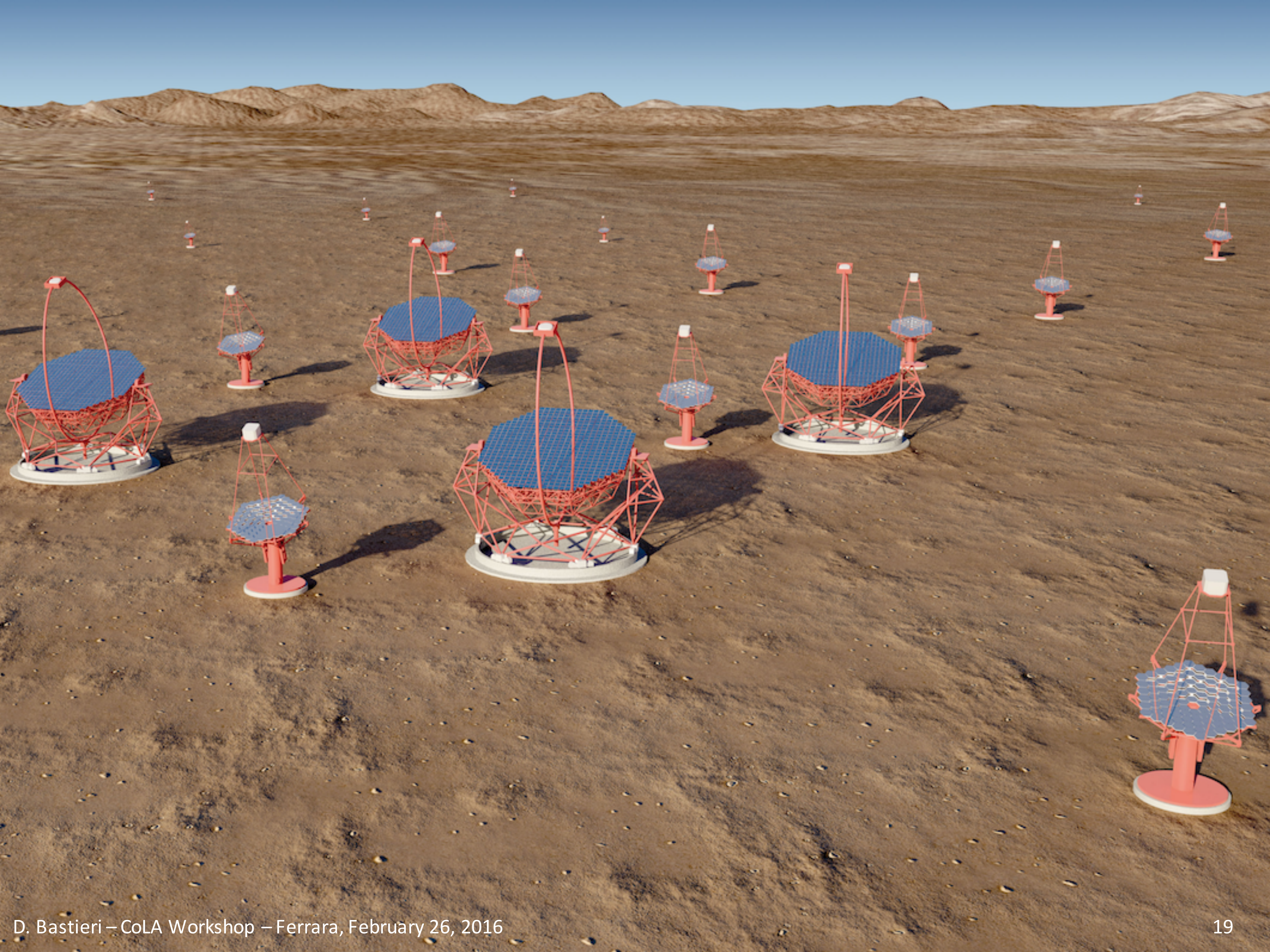
D. Bastieri - CTA Consortium Meeting, Warsaw, 24 September 2013 13/18

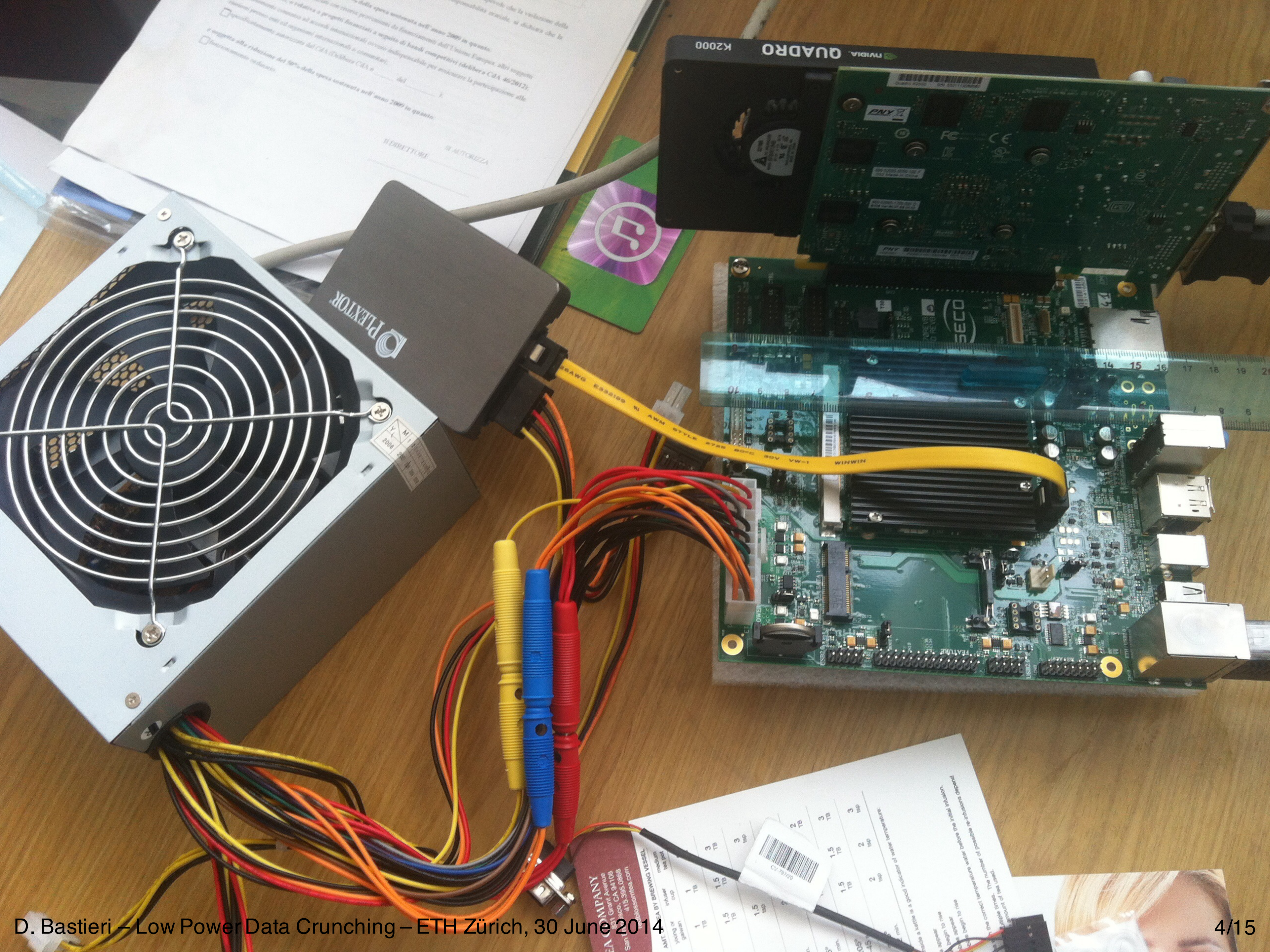
MLA
&LMA

NVIDIA

GPU RESEARCH CENTER

- Maximize the likelihood, given the data
- How to reduce CPU↔GPU data transfer?
- Levenberg-Marquardt vs. MINUIT
  see also de Naurois & Rolland arXiv:0907.2610
- Minimizer resident in GPU memory




QUADRO
K2000
PLEXTOR
SECO
WINWIN

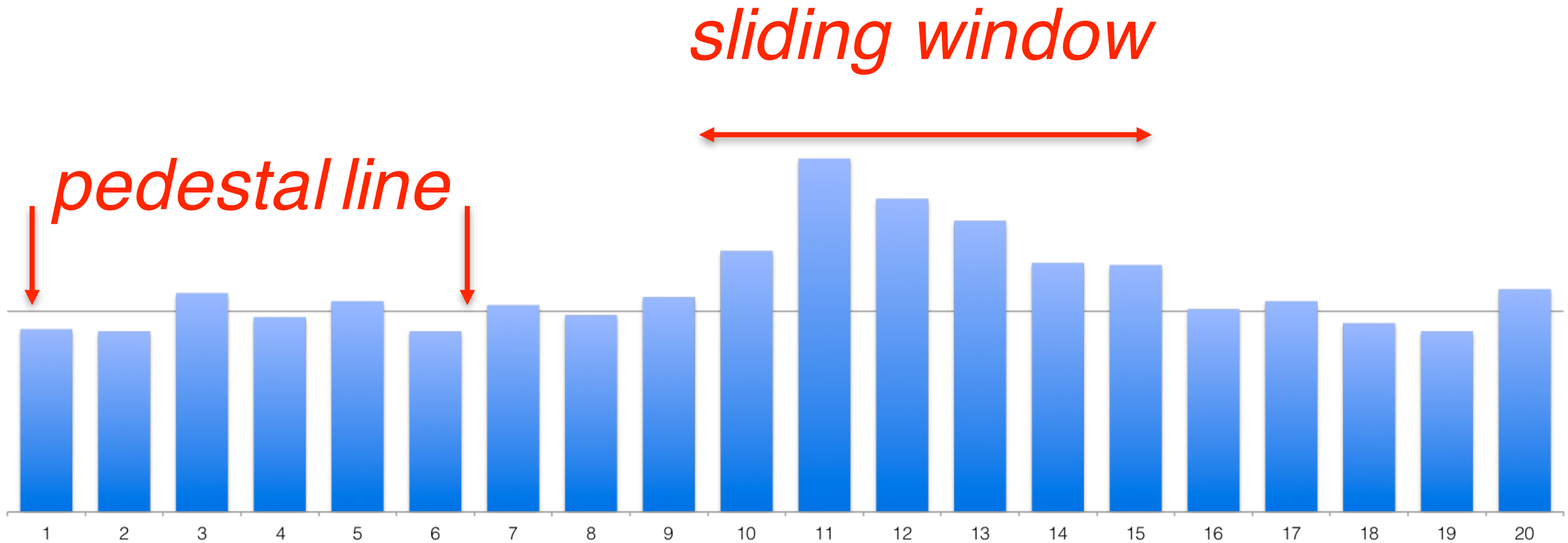
Data Crunching: recipe from OAR

1) Evaluate pedestal offsets from 2k random events
2) Real data input (2GB = 50 s on MAGIC II @200Hz)
3) Pedestal subtraction
4) signal integration via sliding window (short[] → int)
5) ADC counts (int) → (× calibration) → phe (float)
6) phe sorting/clustering/cleaning
7) evaluation of first 10 momenta
8) data output

D. Bastieri & S. Buson (UNIPD)
L.A. Antonelli, D. Gasparrini, S. Lombardi, F. Lucarelli & M. Perri (OAR)

pedestal/calib

1) Evaluate pedestal offsets from 2k random events
⇒ no impact on overall timing of the data crunching.
3) Pedestal subtraction
4) signal integration via sliding window (short[] → int)
5) ADC counts (int) → (× calibration) → phe (float)



ped/cal timing

4) Integrate over the *sliding window* (short[] → (int) → float)
3+5) exploit fma.s $0 = $0×$1+$2

Virtually no difference in timing between pedestal subtraction and pedestal subtraction + conversion [cts → phe]

Typically 25-30 s.

END OF THE TIME BUDGET!

clustering on K2000

Transfer CPU <-> GPU dominates: O(30s), but

30s-budget can accommodate also for:

0) ped/cal +

1) pxl sorting

2) set hi threshold

3) check NN

   > lo/thresh

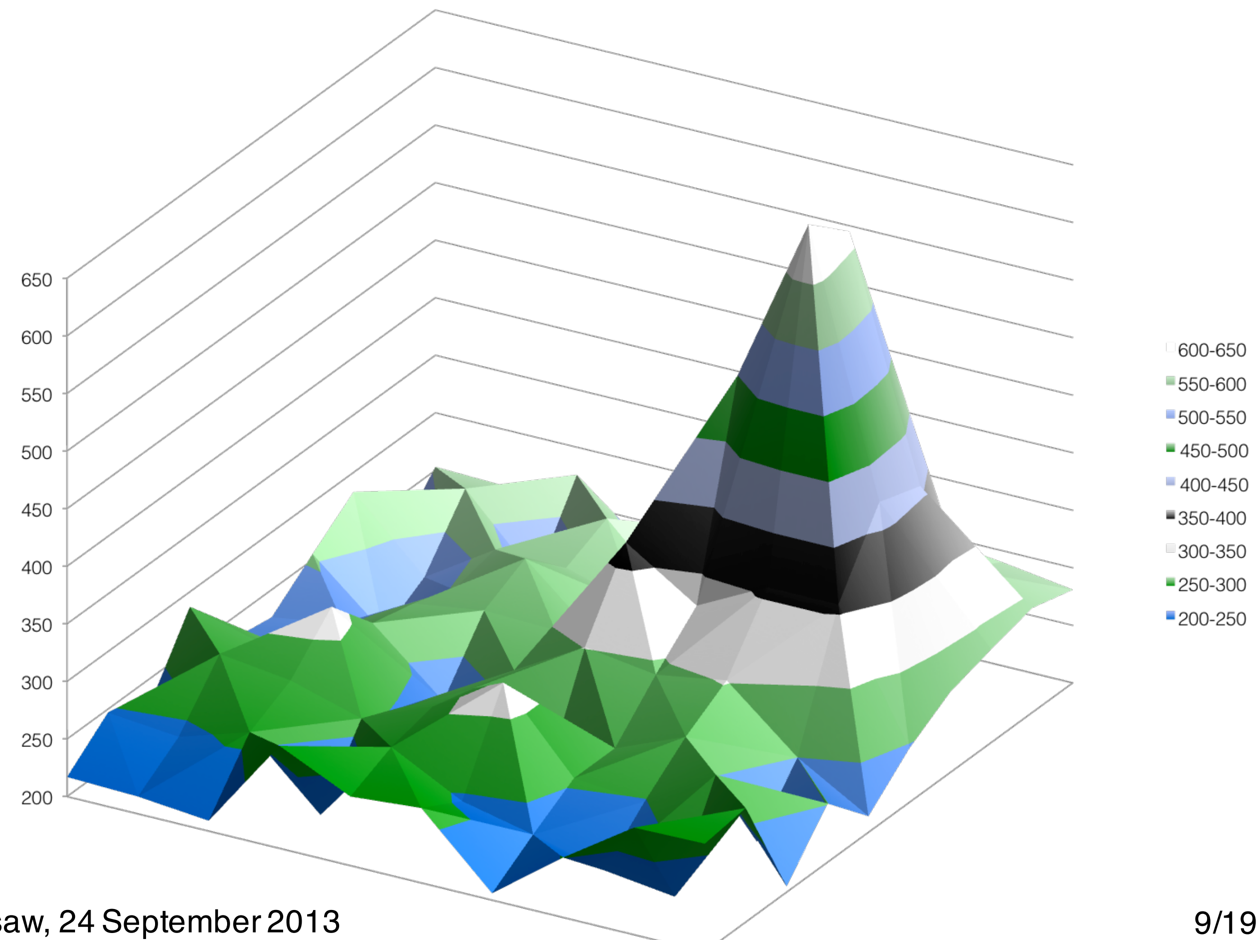
4) else at zero

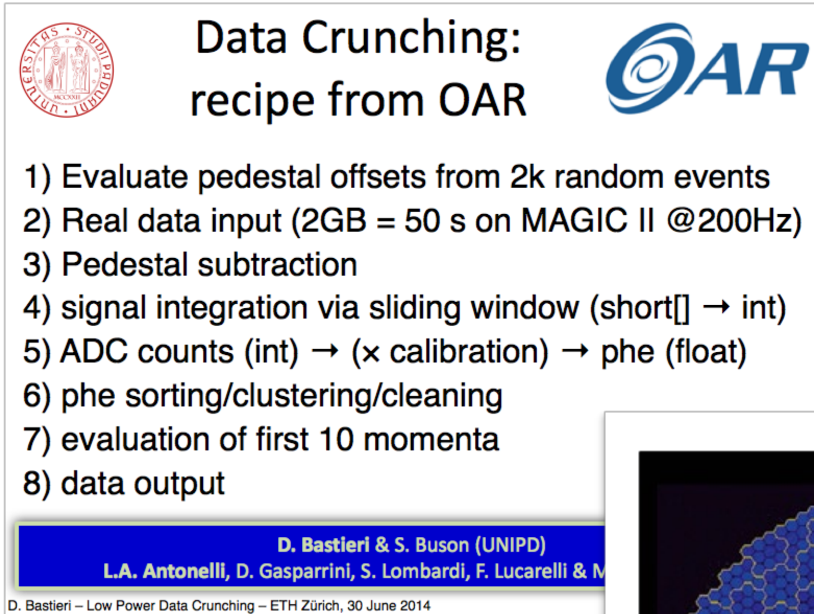
5) evaluate first

   10 momenta

12V rail ≤ 1.6A

=> P ≤ 20W



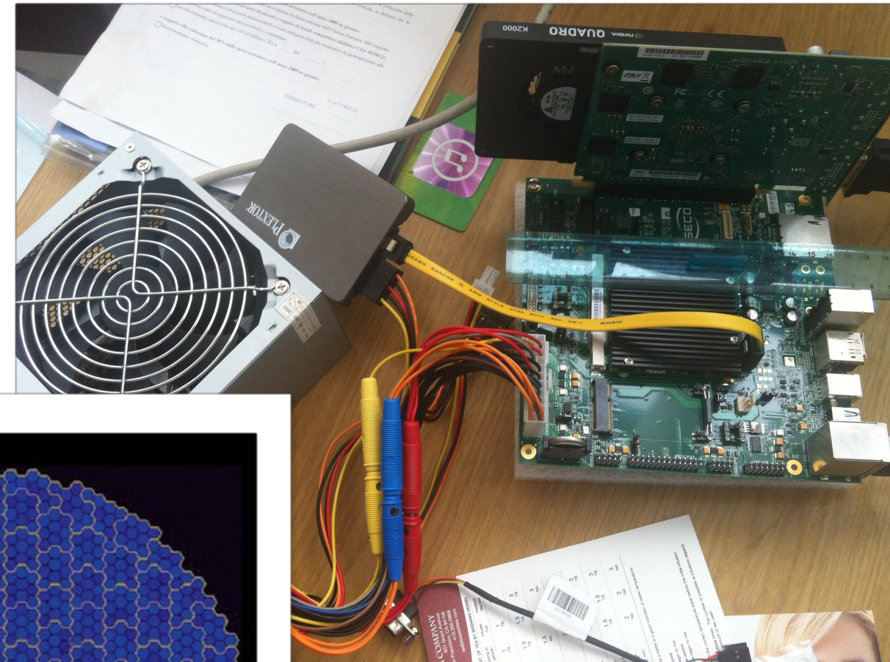


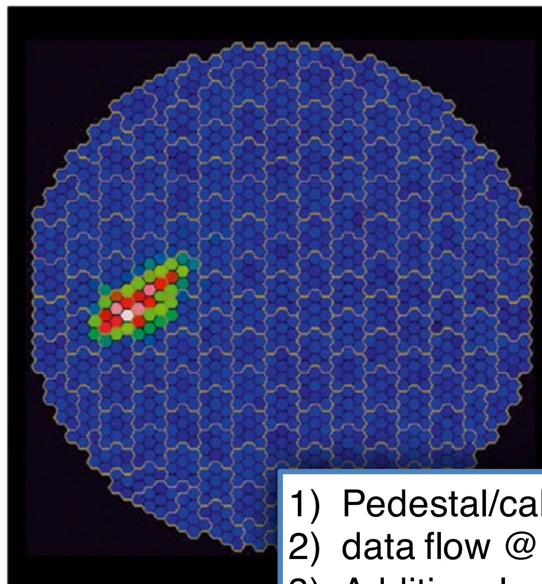
Data Crunching: recipe from OAR

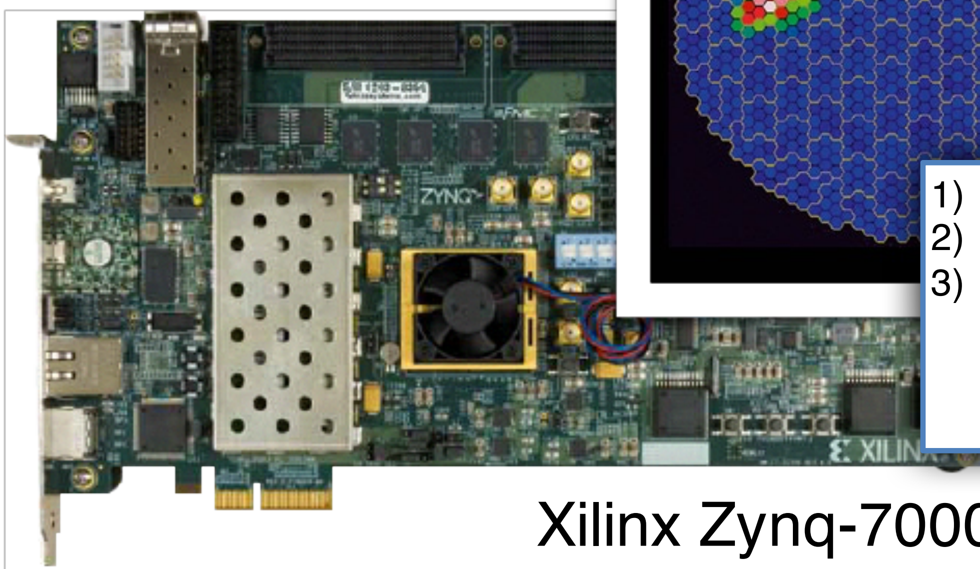
1) Evaluate pedestal offsets from 2k random events
2) Real data input (2GB = 50 s on MAGIC II @200Hz)
3) Pedestal subtraction
4) signal integration via sliding window (short[] → int)
5) ADC counts (int) → (× calibration) → phe (float)
6) phe sorting/clustering/cleaning
7) evaluation of first 10 momenta
8) data output

D. Bastieri & S. Buson (UNIPD)
L.A. Antonelli, D. Gasparrini, S. Lombardi, F. Lucarelli & M

D. Bastieri – Low Power Data Crunching – ETH Zürich, 30 June 2014





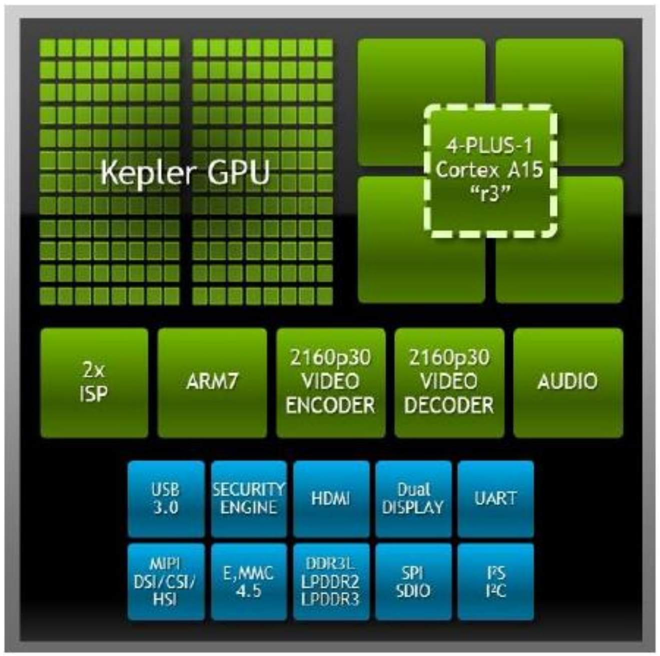


Xilinx Zynq-7000

1) Pedestal/calibration feasible on ARM @5W.
2) data flow @1Gb/s, data processing @~2GB/min
3) Additional analysis:
   a) spawn it to GPU's cores (add 20W or <*P*> ~11W)
   b) filter through FPGA (?2W? <*P*> ~8W)
   c) try out Jetson-TK1 (SoC)

NVIDIA Jetson TK1

- Heterogeneous System-on-Chip
- CPU: Quad-core ARM A15
- GPU: Kepler architecture - 1 Multiprocessor
- RAM: 2GB (unified address memory)
- OS: Ubuntu 14.04 Linux for Tegra (L4T)
- CUDA 6.5
- I/O: SATA 3Gb/s HDD (no on-board eMMC)



Average power consumption: < 10 W

ASTRI Pixel-level algorithms easily express parallelism

- **Calibration**

  Essentially an *embarrassingly parallel*, Fused Multiply-Add operation (ASTRI camera outputs integrated ADC counts):

  PHE = ADC * coefficient + pedestal

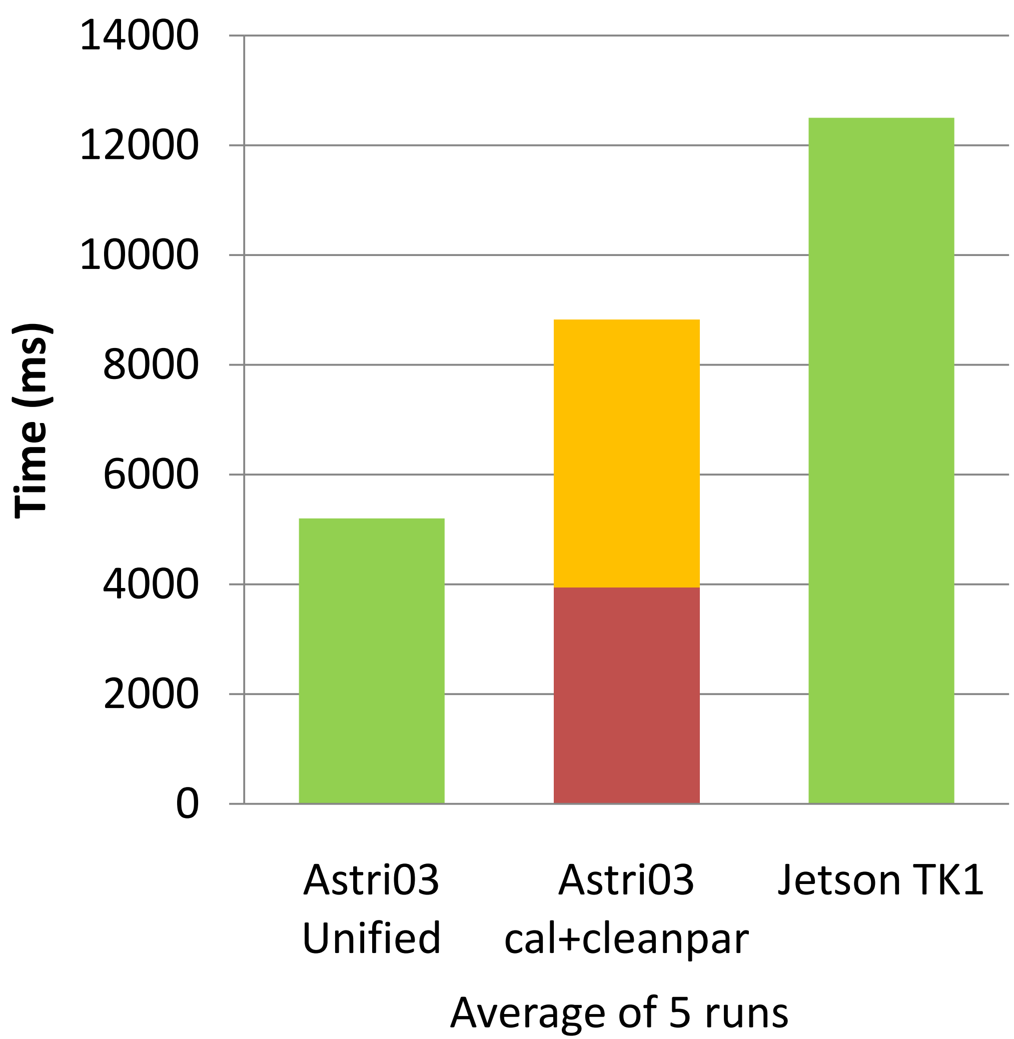
  `$0 = $0×$1+$2`

- **Cleaning**

  Two pass cleaning (two threshold comparisons)

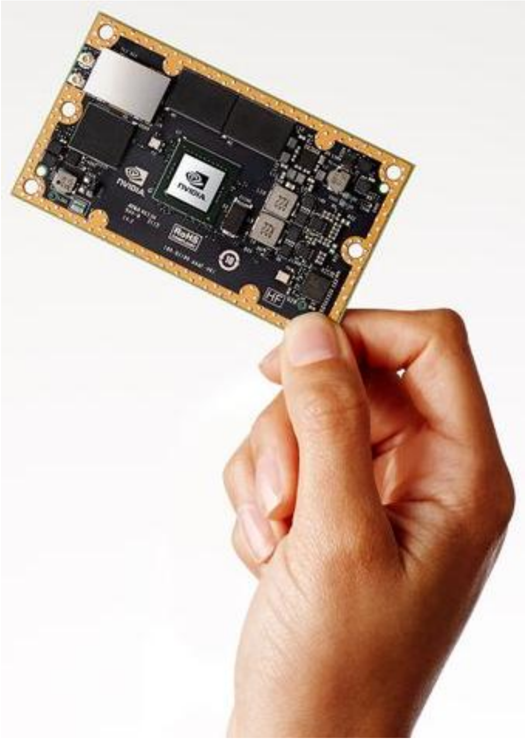
  Well suited to parallelism

Low-power Unified module



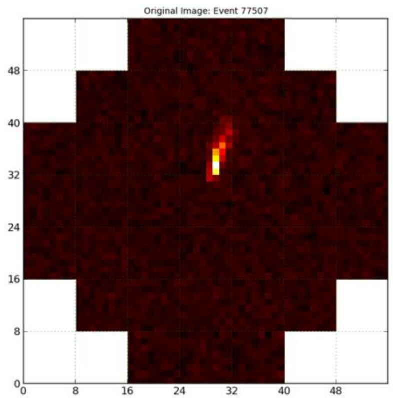
- Processing from DL0 to DL1b (size-reduced telescope-wise data)
- All done in 12.5s:
  4400 evt/s
  > 4x peak acquisition rate
- 2.5x slower than server UM
  1.4x slower than separate modules
  30x less power
- Still plenty of time left for online analysis!

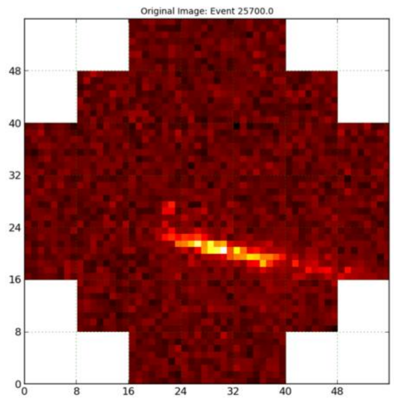
NVIDIA Jetson TX1

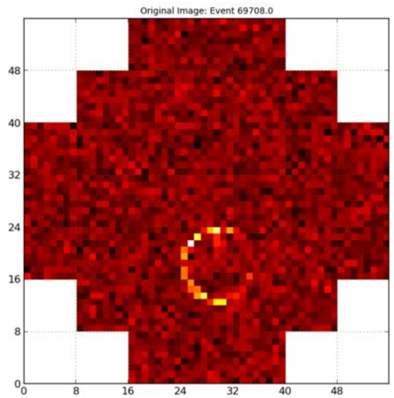


- Latest generation embedded module from NVIDIA (announced Nov. 11th 2015)
- Credit-card size, touted of same ≈10W consumption (max 15W)
- CPU: Quad-core ARM A57
- GPU: 256-core Maxwell arch (2 SMM multiprocessors)
- 4GB RAM, Gigabit Ethernet
- Devkit with carrier board: $600

Reference Test Case

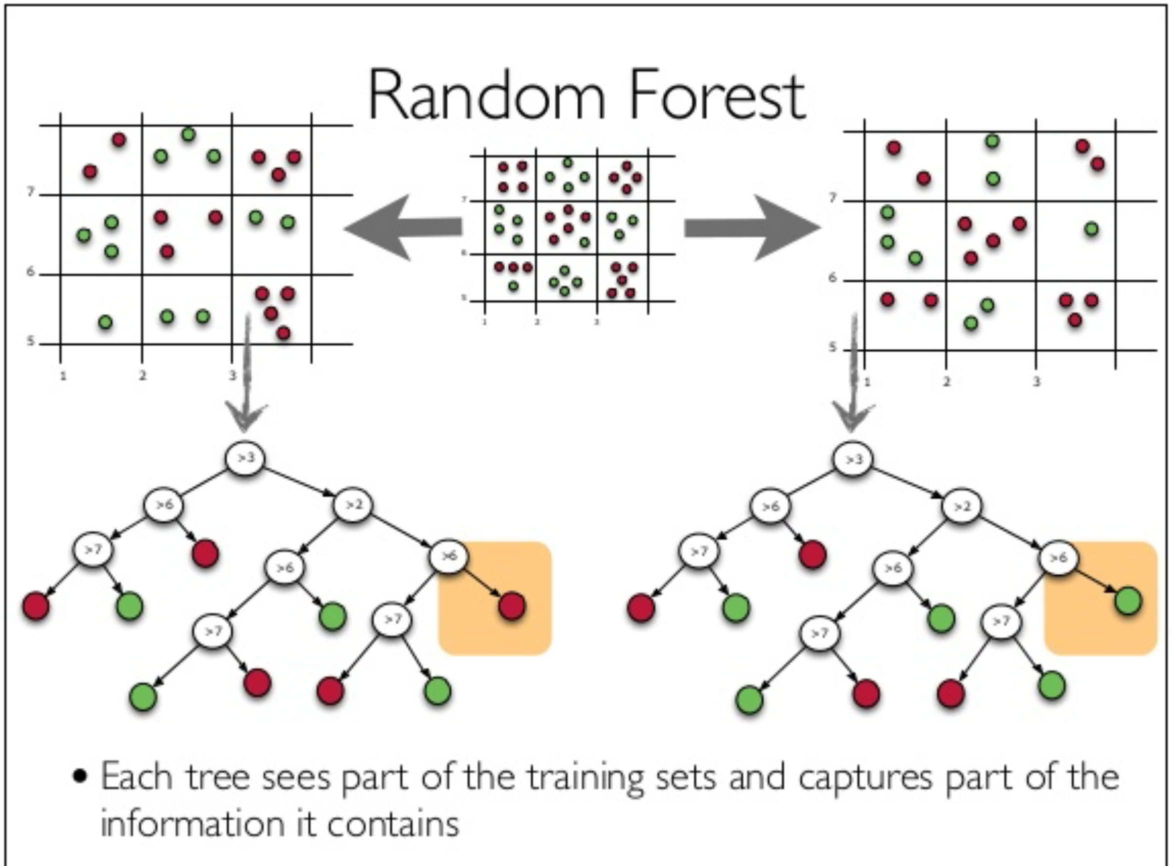






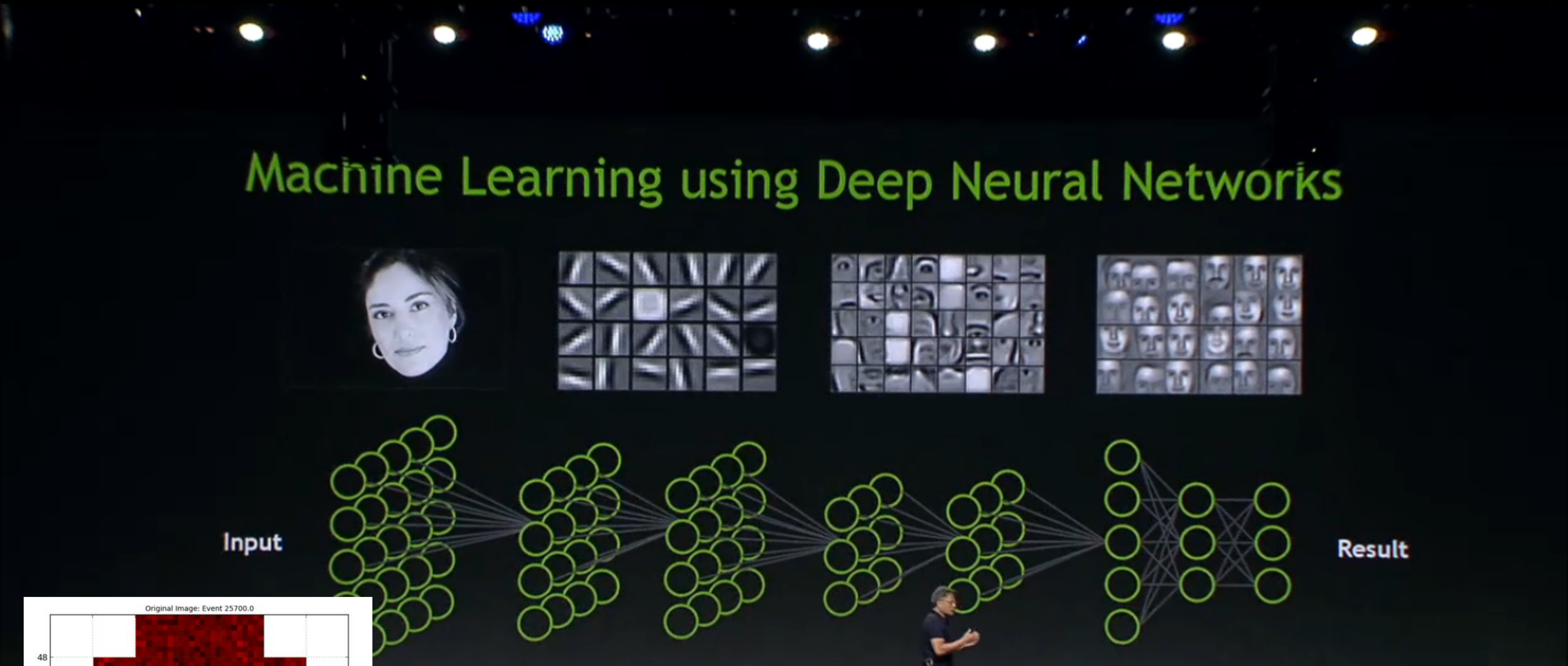
- 500MB (= 55049 events) of simulated DL0 “real data”
- ≈ 110s of nominal acquisition rate (500Hz)
- ≈ 55s of projected peak rate (1000Hz)
- ≈ 80.5% of events survives pruning with default settings
- Compliant with format and size agreed with camera hardware team

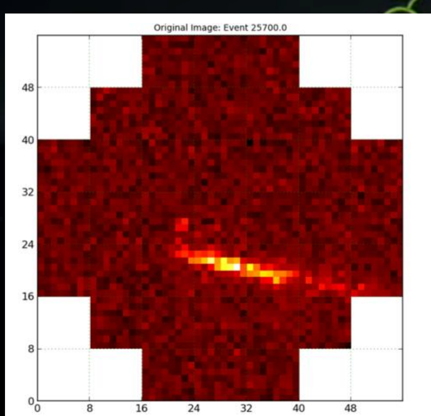
What’s next?



Hadronness
Energy estimation
Incoming direction

What's next? DNN!



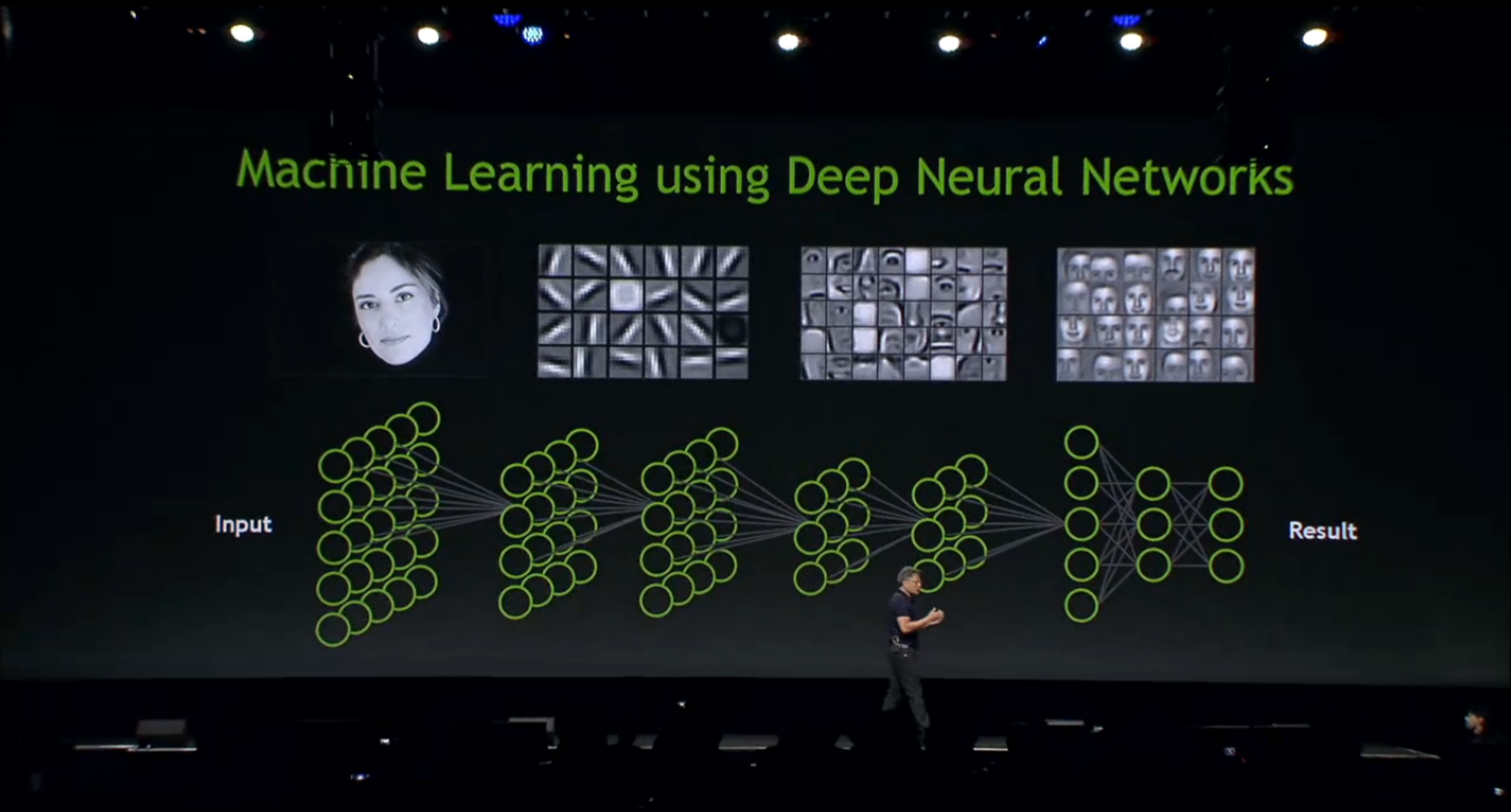


Hadronness = 35%
Energy = 565 GeV
dir: RA=$19^h58.4^m$
dec=35°12.1'

Conclusion

- Gamma-Ray Astronomy is an optimal test-ground for Low-Power Computing and High-Throughput Computing.
- Gamma-Ray Astronomy from Space needs a lot of computing power
  - Mainly images or *sparse* matrices: data parallelism!
  - GPU are ideal to speed up execution!
  - Still trying to find a *resident* minimizer
- Gamma-Ray Astronomy from ground needs a lot of computing power
  - Mostly in the realm of HTC (calibration, cleaning, image momenta…)
  - Calibrations may be done with ARM
  - Calibrations may be done with FPGA (lower Watts, but worth the additional burden?)
  - Additional analysis are feasible on NVIDIA Jetson T*1
- Where to go next for Gamma-Ray Astronomy?
  - What about DNN?
  - Are they good for cleaning and extracting physical information?

What’s next? DNN!



We are recruiting!

Development system

Dual-processor Intel Sandy Bridge @ 2GHz with 16 physical cores and 128GB of RAM (8GB per core)

GPU gen3 READY and n.1 installed (up to 3 GPU drives)

8 disk slots of 4TB each (to export 2 different redundant drives of ~12TB each)

direct link and share with the storage system

- Installed @ OAR Monte Porzio Catone
- Accelerator: NVIDIA Tesla K20c (20-30% slower than K40)

MaxEnt 1985
(6-mon proc)

MaxEnt
2014
(6-sec proc)