High Performance Computing for gamma ray detection

22/09/2016

Pierre Aubert, Jean Jacquemier, Thomas Vuilllaume, Florian Gaté, Gilles Maurin, Nahid Emad, Armand Fiasson, Giovanni Lamanna

Pierre Aubert

Versailles/Saclay

- Nahid Emad
- LI-PaRAD
- Maison de la simulation

LAPP

- Gilles Maurin
- Jean Jacquemier
- Thomas Vuilllaume
- Florian Gaté

CTA

- Data analysis

ASTERICS

- CTA
- SKA
- LSST
- KM3Net
- E-ELT
- LIGO

Transversal Projects

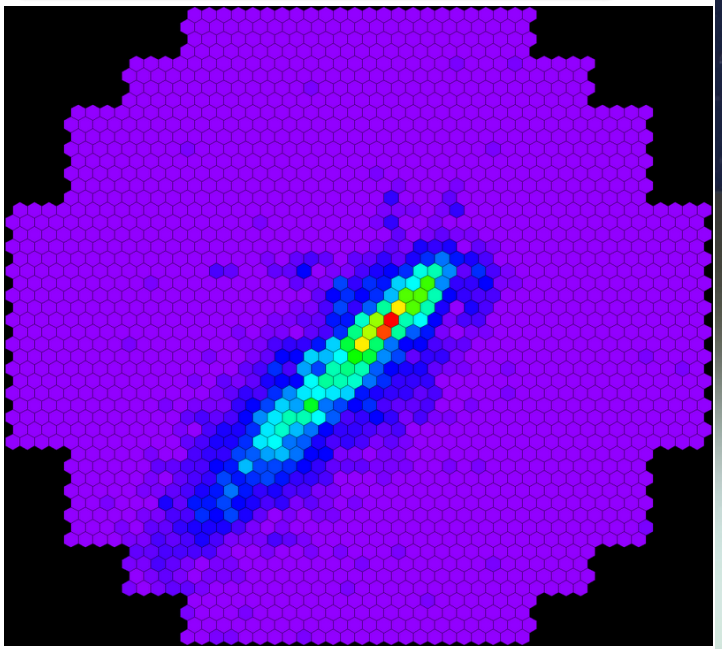
LAPP

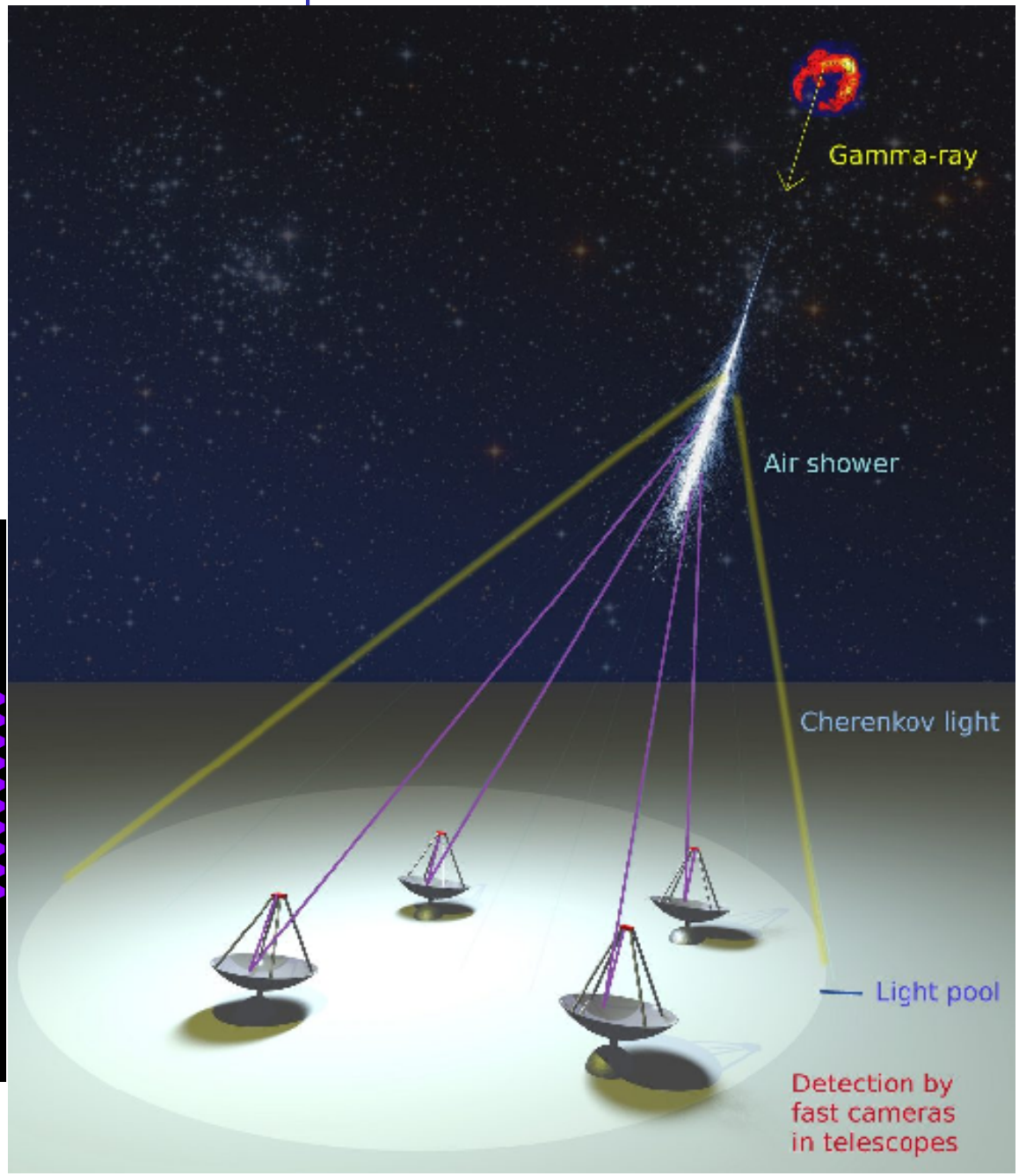
- ASTERICS
  - CTA
  - LSST
- COMUE IDEX UGA
  - Call for project Pôle-PAGE
  - Data@UGA project

Detection of particle showers in the atmosphere

Cosmic rays

- 89% : protons
- 10% : heavy nuclei
- 1% : electrons
- 1 $^{0}/_{00}$: gamma rays





The H.E.S.S. experiment (Namibia) (170 scientists)

Telescopes

- 4 small, 13 m of diameter
- 1 large, 28 m of diameter
- 2 types of camera

Data flow

- 200 frames per second
- 3 MB/s

Monthly time

- 100 h observation
- 1000 h analysis

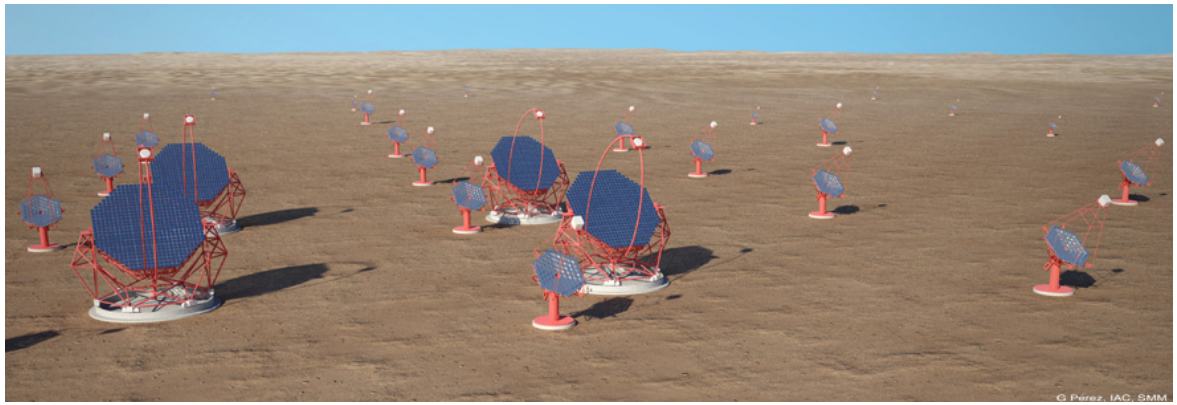
The international CTA project (La Palma, Chili) (1200 people)

Observatory

- Provide/Store raw data to the international community
- Create/Maintain/Provide analysis to the international community
- Accessible via a web portal

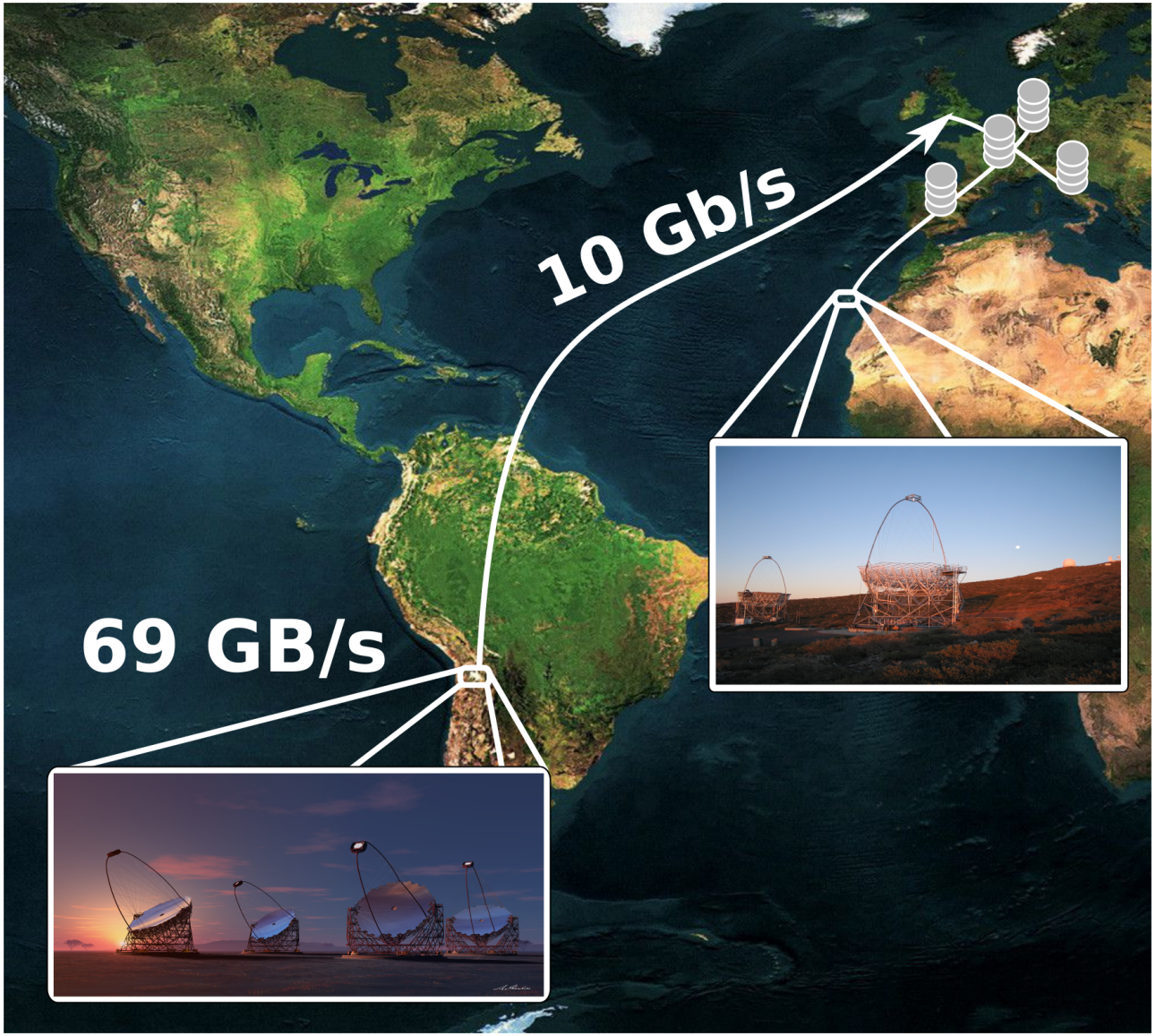
Telescopes

- 126 telescopes
- 3 telescope sizes
- 7 camera types

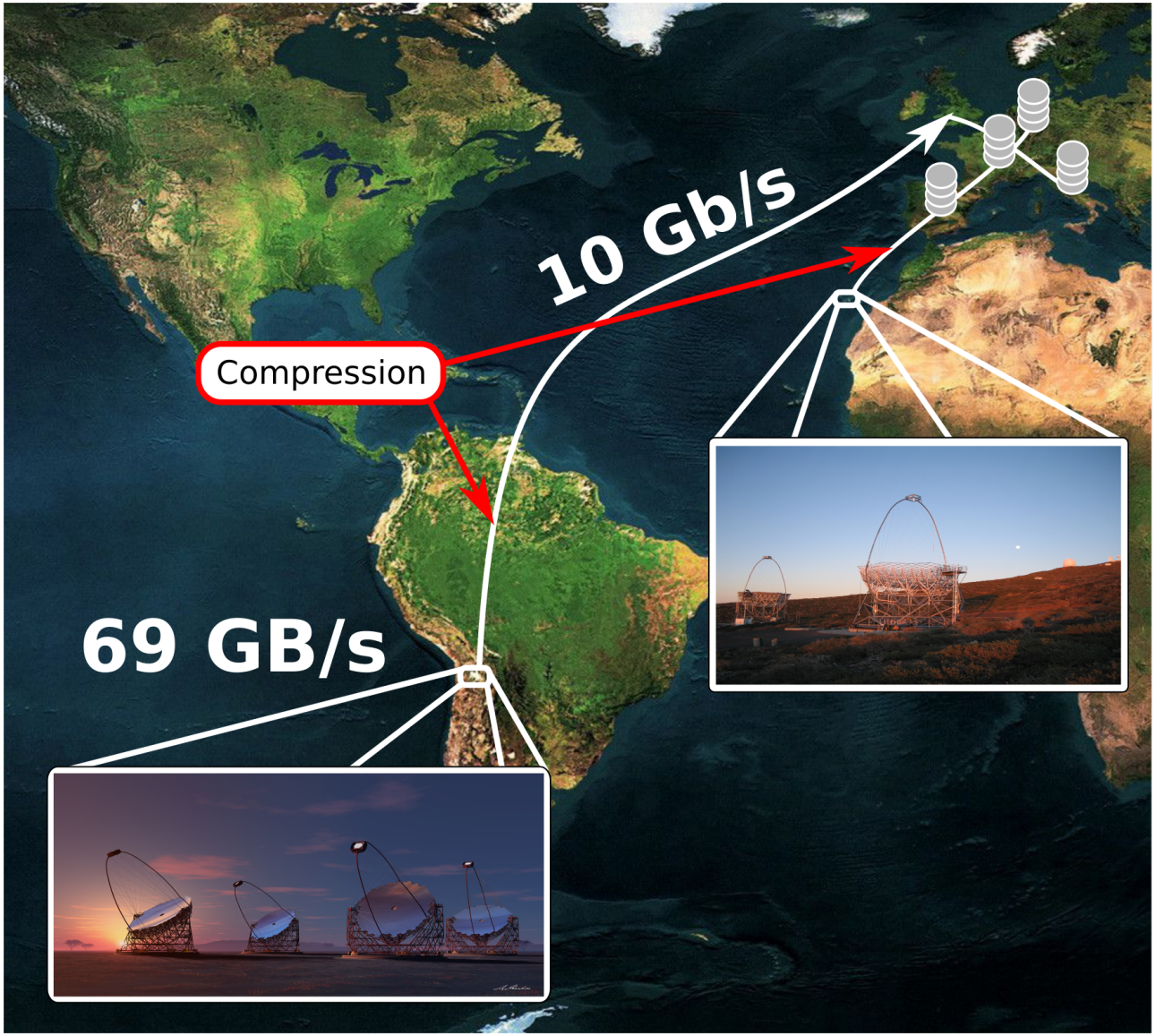


Data rate

- 15 000 frames per second per large telescope
- 69 GB/s $\Longrightarrow$ 2 PB/yr
- Optic fiber line : 10 Gb/s
  - Because 100 Gb/s line $\Longrightarrow$ 23M€ per year


10 Gb/s
69 GB/s

First challenge : the Data compression



Raw data Compression Ratio and speed (Lossless compression)

Test file run 497 (Paranal, Gamma Monte-Carlo CTA PROD_3)

- 475 MB
- More than 99% ADC values

	Compression ratio	Time	File size (MB)
LZMA (7z)	4.84	7 min 48.636 s	98
Advanced Polynomial Compression	3.74	3.7 s	127

Raw data Compression Ratio and speed (Lossless compression)

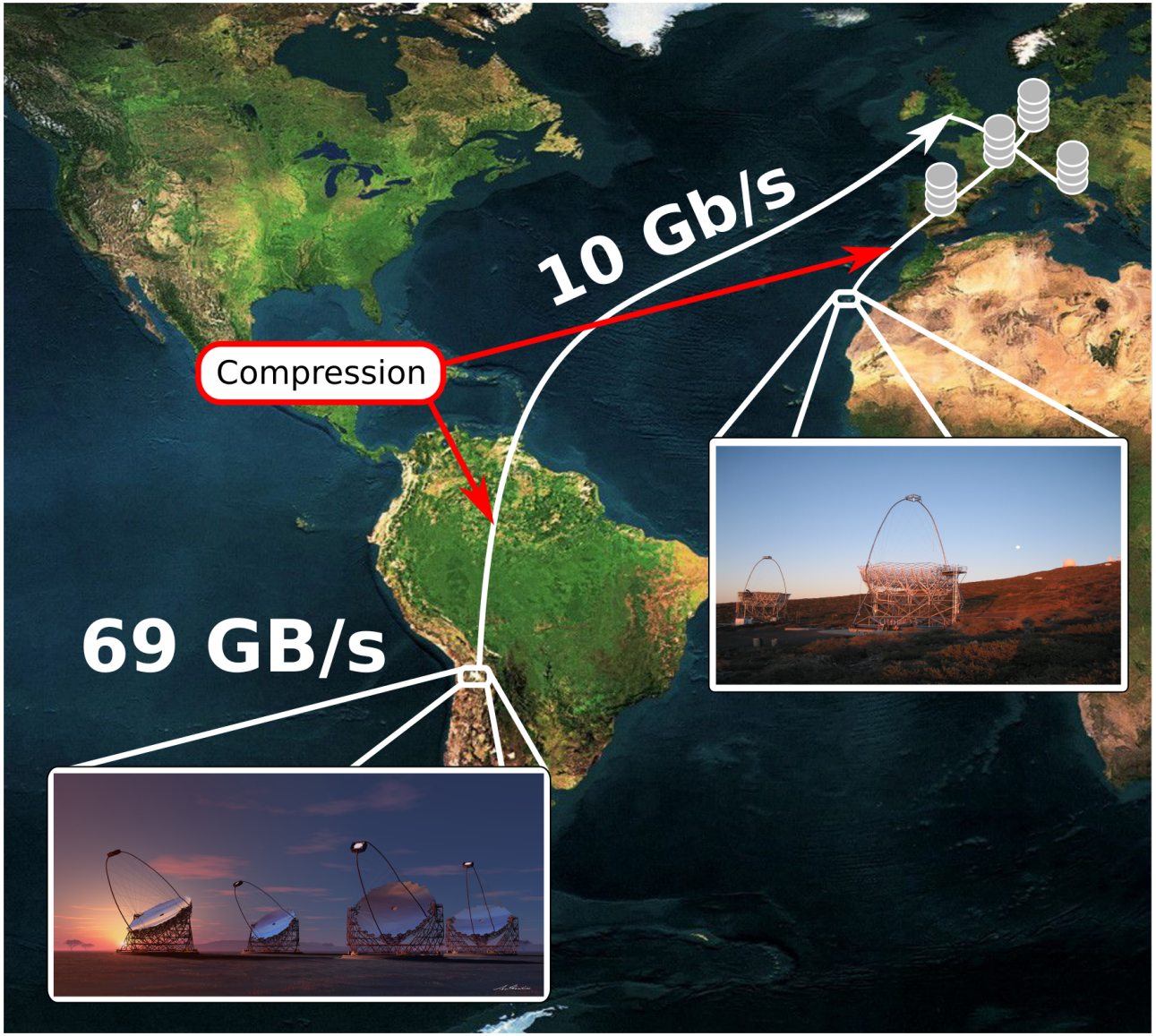
Test file run 497 (Paranal, Gamma Monte-Carlo CTA PROD_3)

- 475 MB
- Up to 99% ADC values

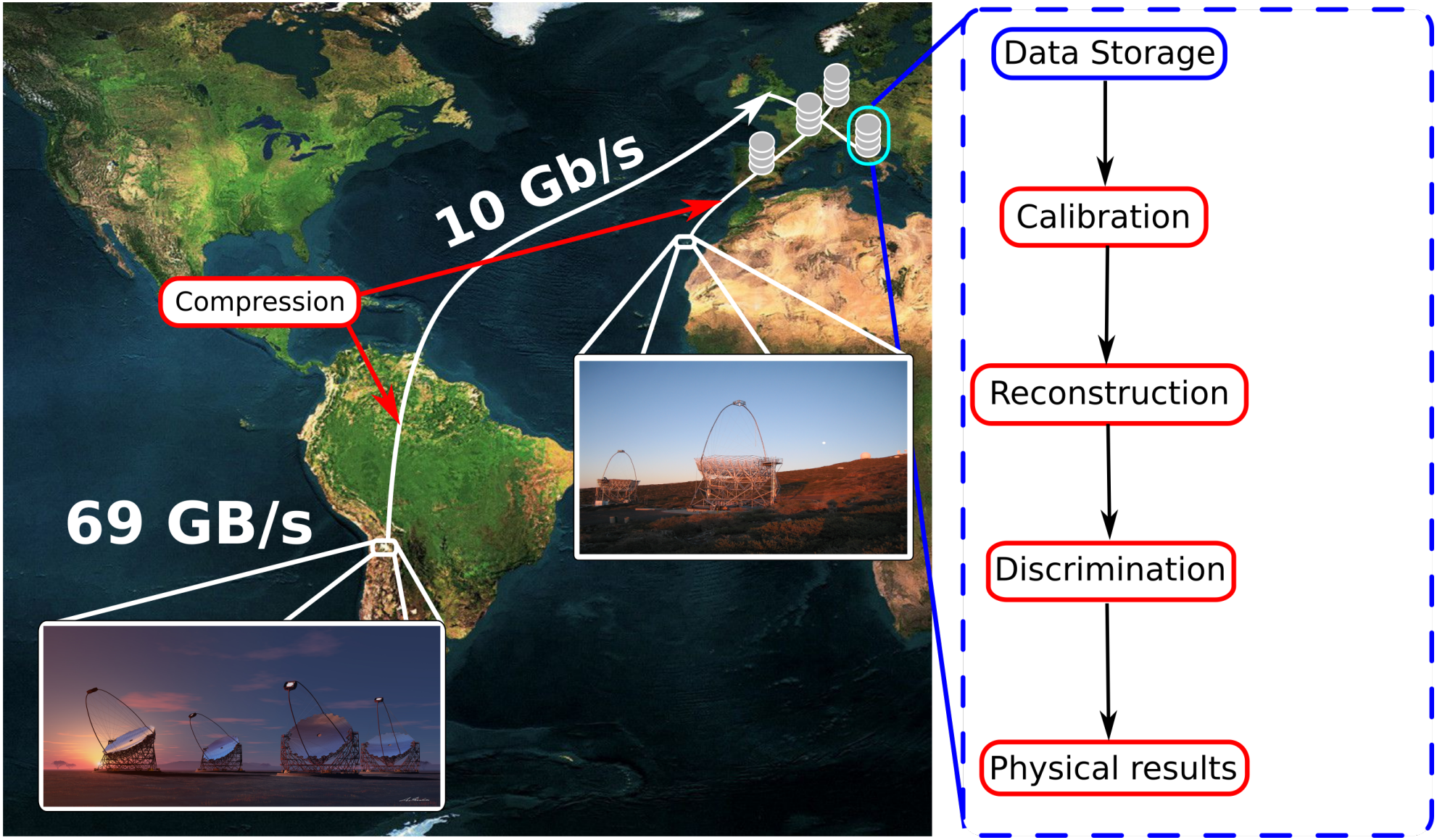
	Compression ratio	Time	File size (MB)
LZMA (7z)	4.84	7 min 48.636 s	98
Advanced Polynomial Compression	3.74	3.7 s	127
Advanced Polynomial Compression + LZMA	**4.84**	**24.646 s**	**98**

Same ratio but faster ($\times 19$) compression !

- Can be used for any data format
- Publication ongoing


10 Gb/s
Compression
69 GB/s

The CTA data analysis



Data format

What is good for an HPC data format ?

- Allows CPU data pre-fetching
  - With contiguous data
  - Guaranty data locality
  - Cache friendly
- Allows vectorization
  - With aligned data for vectorizable computation
- Must be as simple as possible (user friendly)
  - Simple types (float, int, double, unsigned int, ...)
  - Tables
  - Allows the simpler solution for complex problem

LAPP Sorted HPC Autotuned Data Optimized Kernel format

Generated Data format for CTA

Describes :

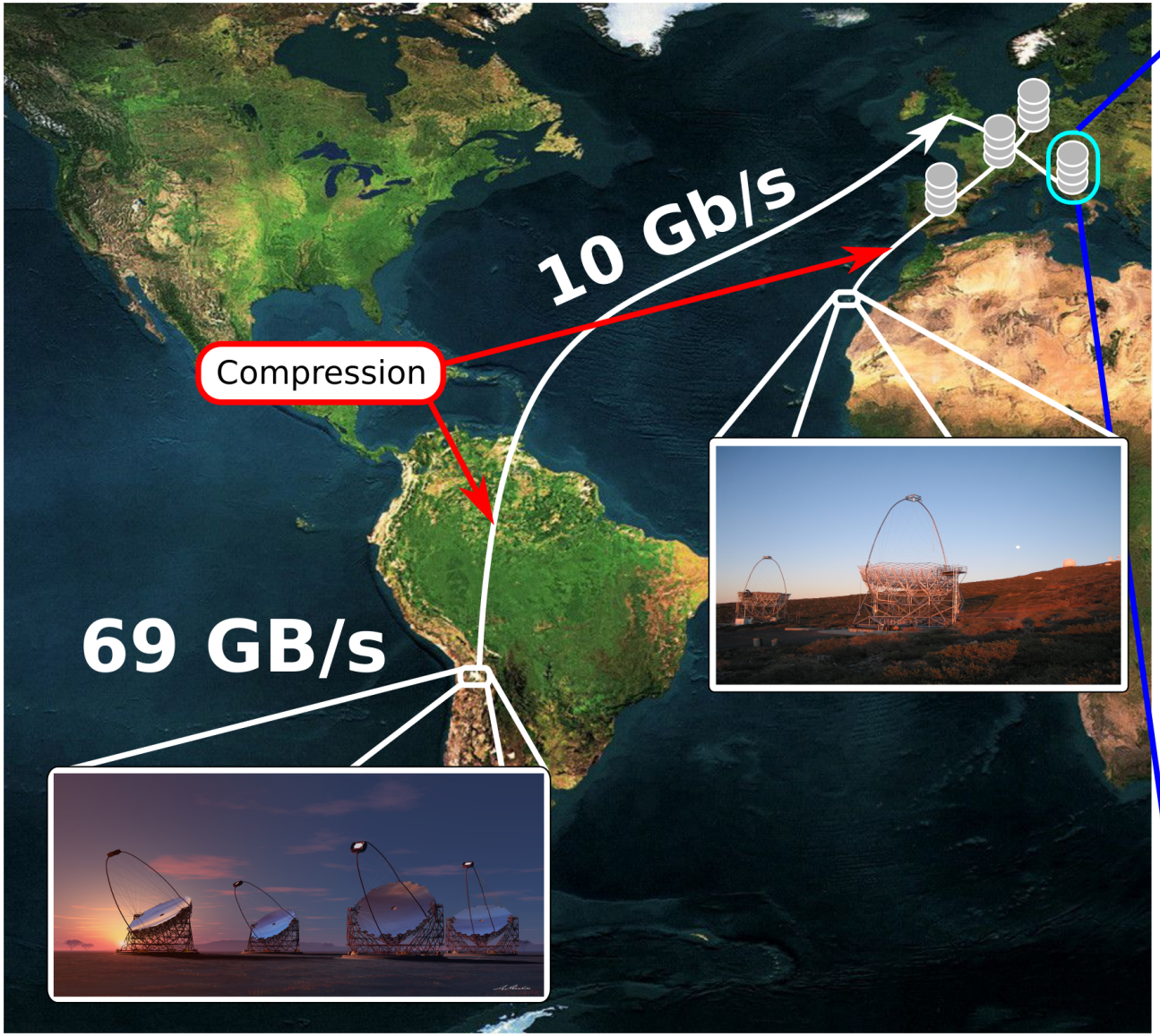
- All the telescopes
- All the Cameras
- The CTA PROD_3 Monte-Carlo
- Allows Layouts and Masks
- DL0, DL1, DL2

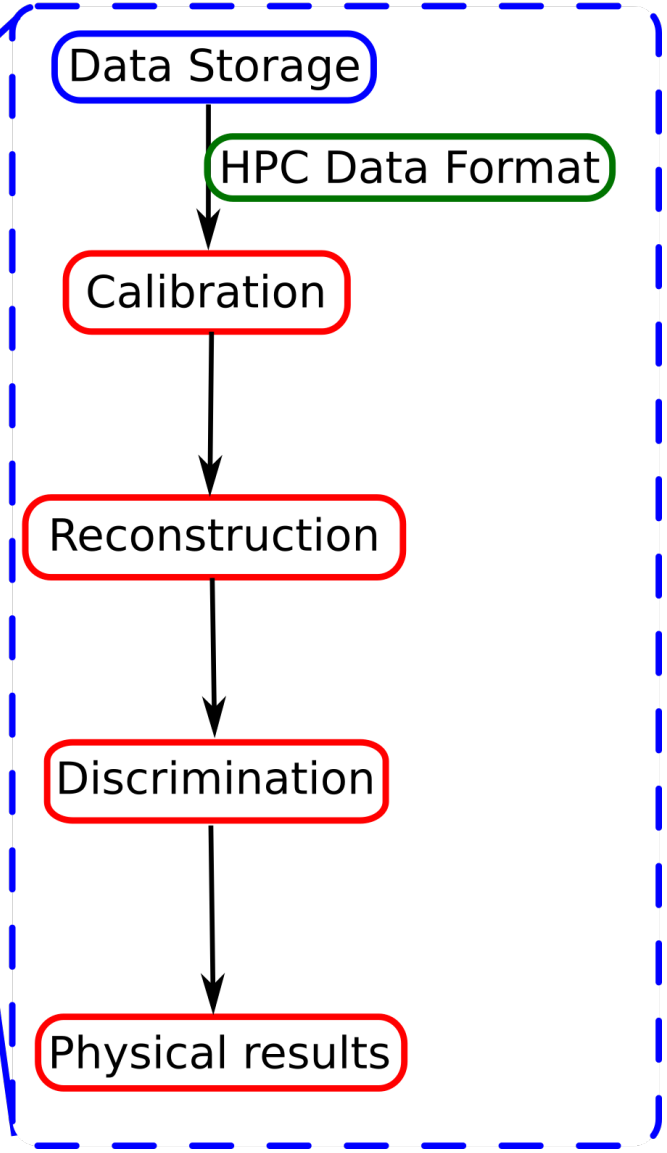
Allows :

- Allows CPU data pre-fetching
  - With contiguous data
  - Guaranty data locality
  - Cache friendly
- Vectorization of algorithms
  - Calibration, Cleaning, Hillas

And it's user friendly

Second challenge : the data analysis





Second challenge : the data analysis

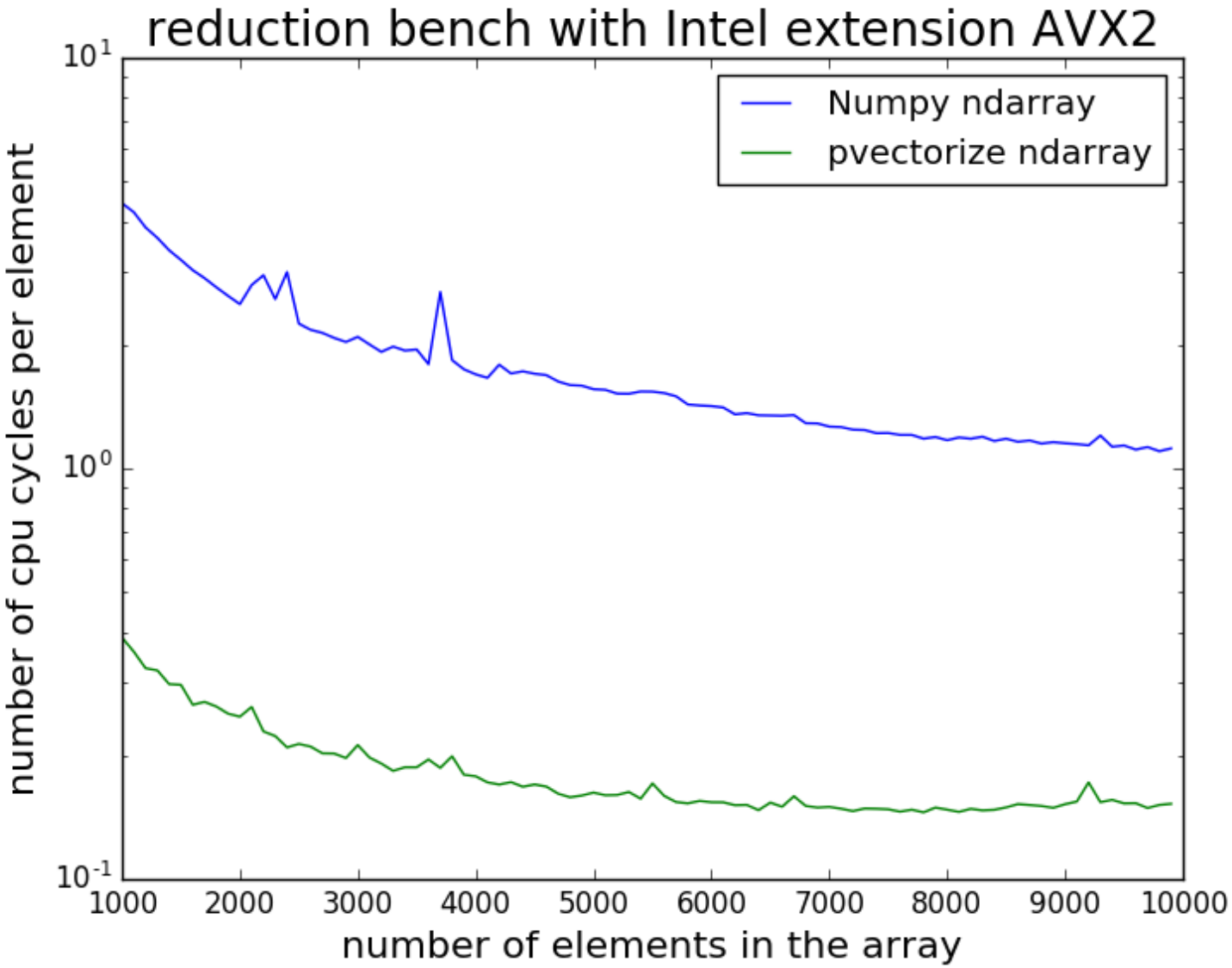
Programming language for CTA

- Python
  - Good to develop tests
  - Bad to analyse data in production

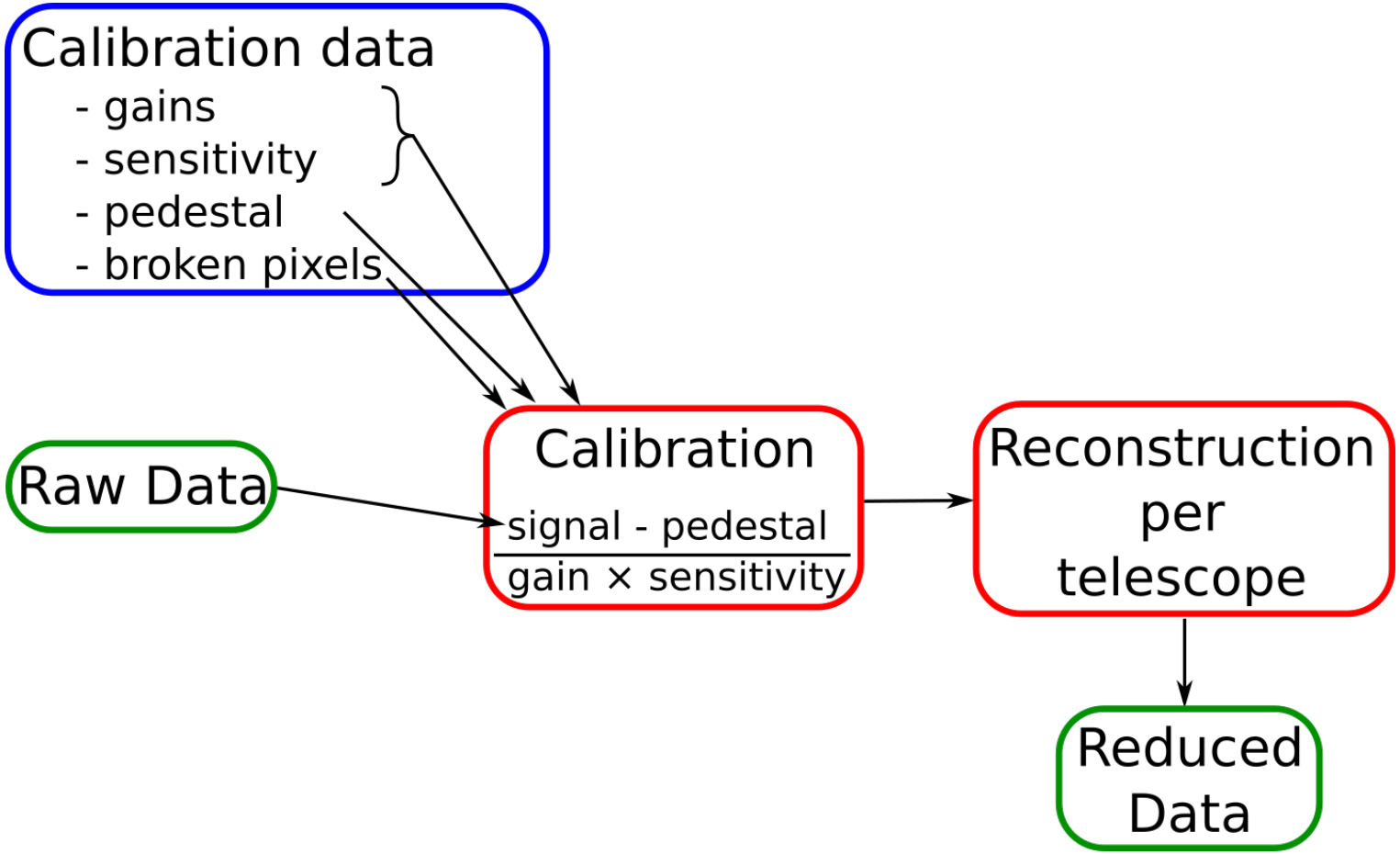
Optimization

- Fast C++ library
- Wrapper python

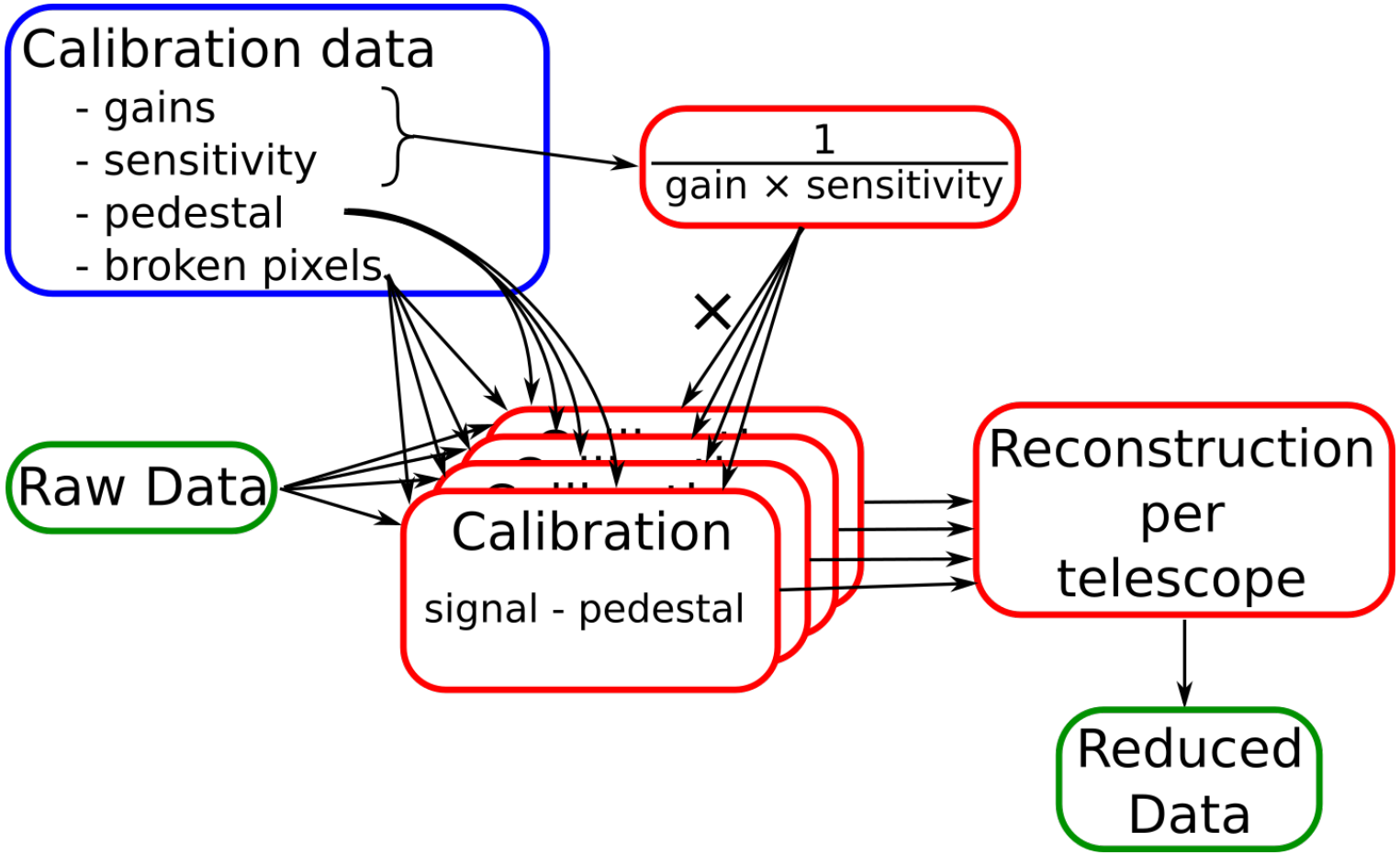
Second challenge : the data analysis



Raw data calibration



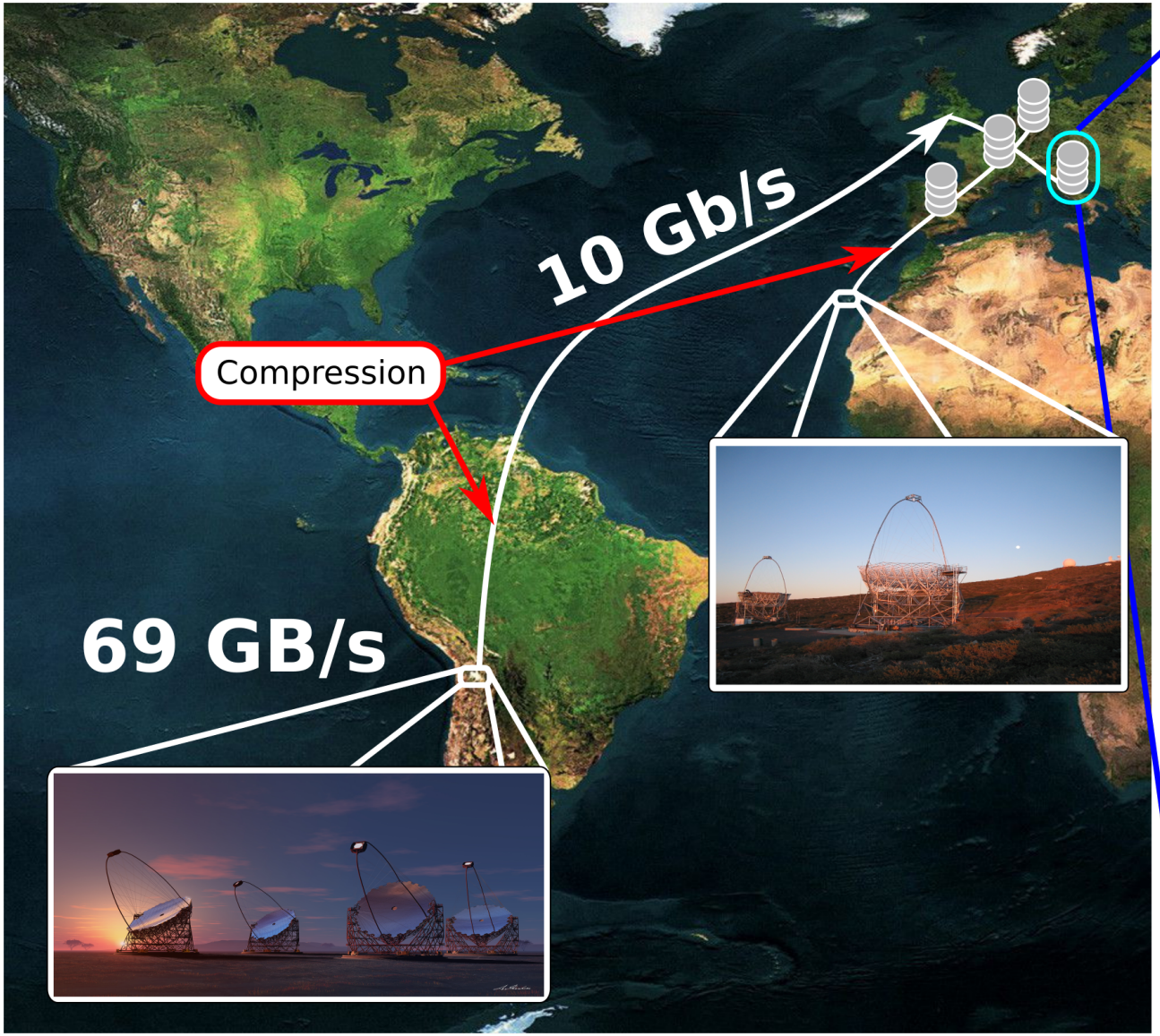
Raw data calibration

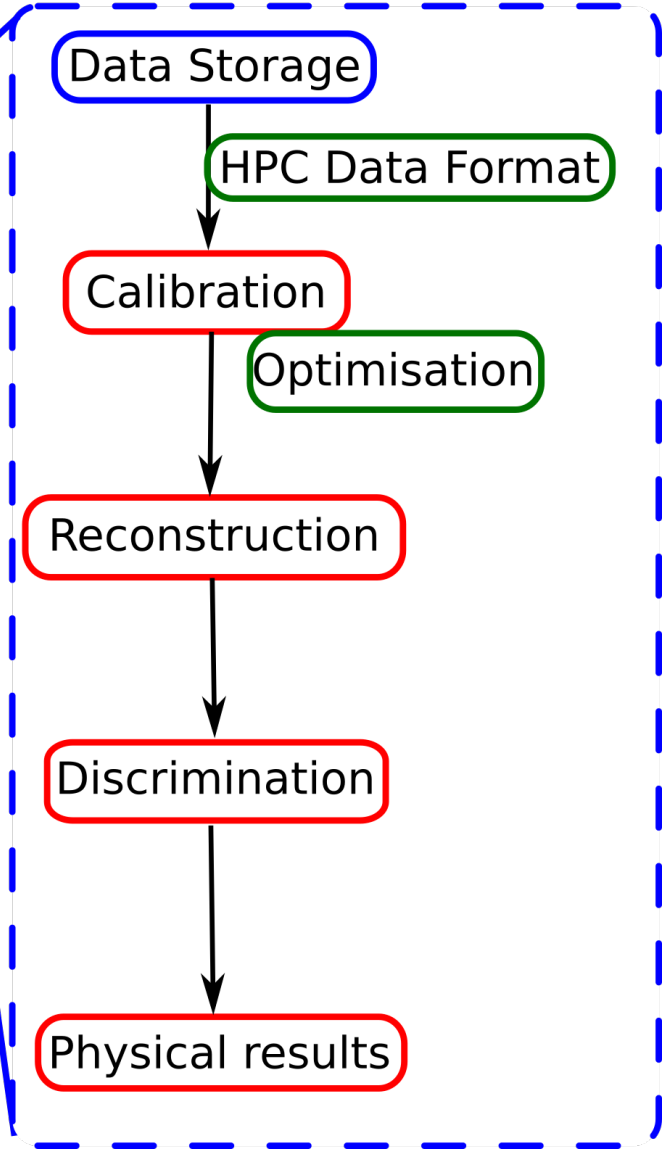


Optimization

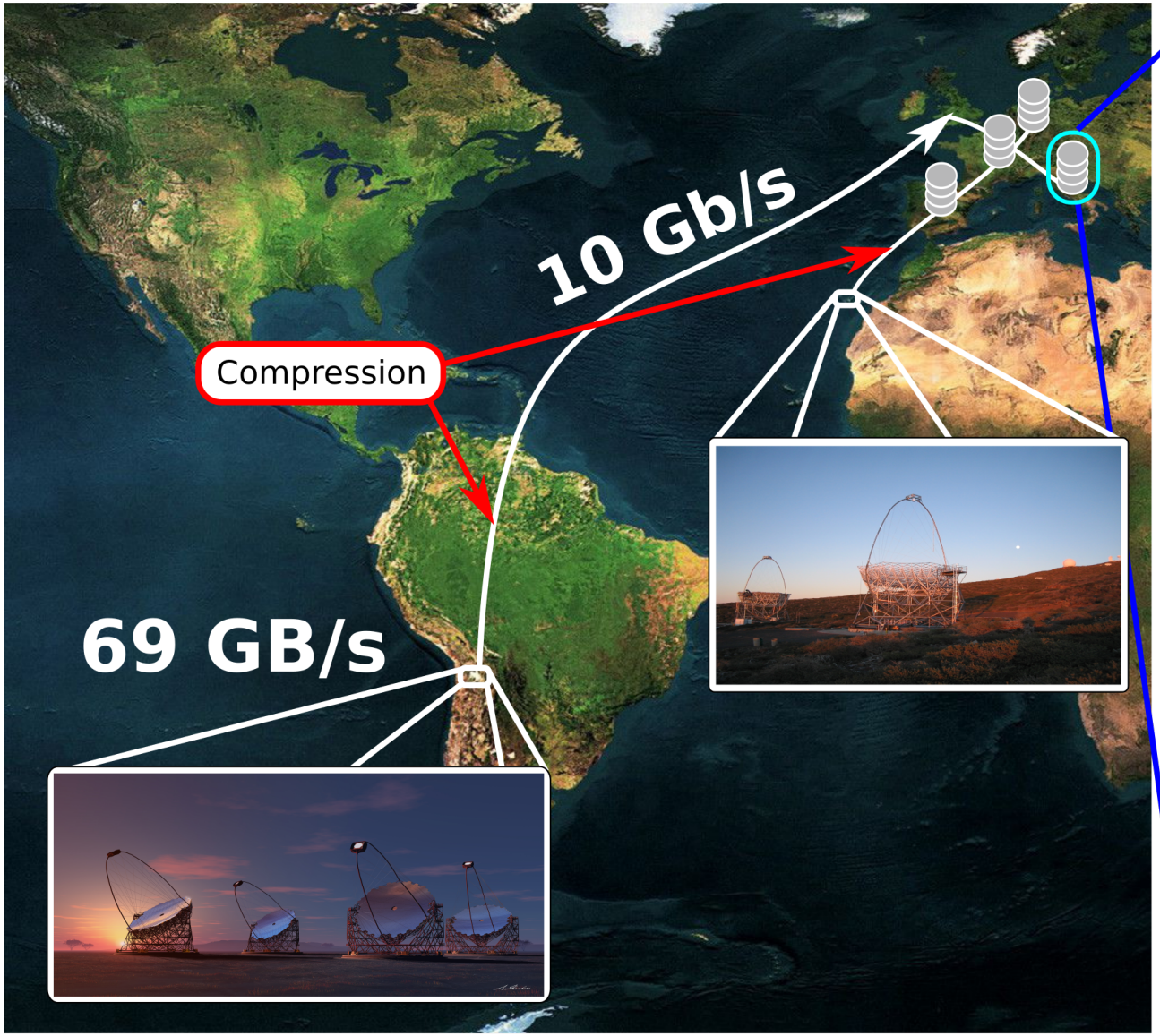
- Clearly vectorizable (contiguous data, no back dependencies)
- Parallelization non efficient because the pictures are too small

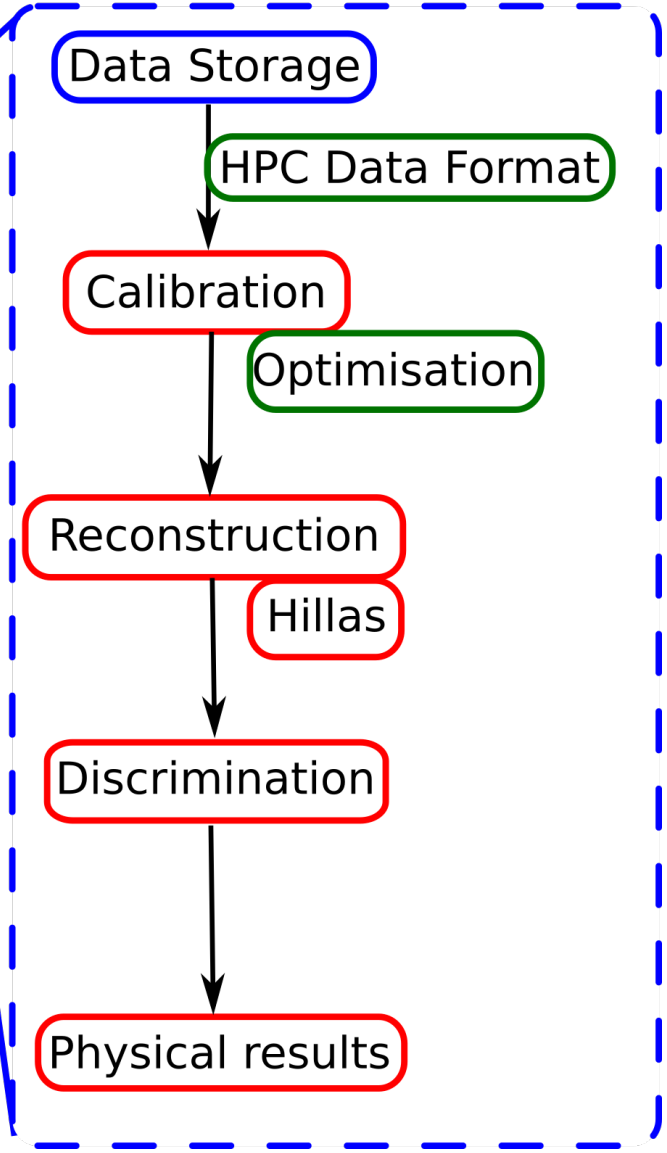
Second challenge : the data analysis





Second challenge : the data analysis

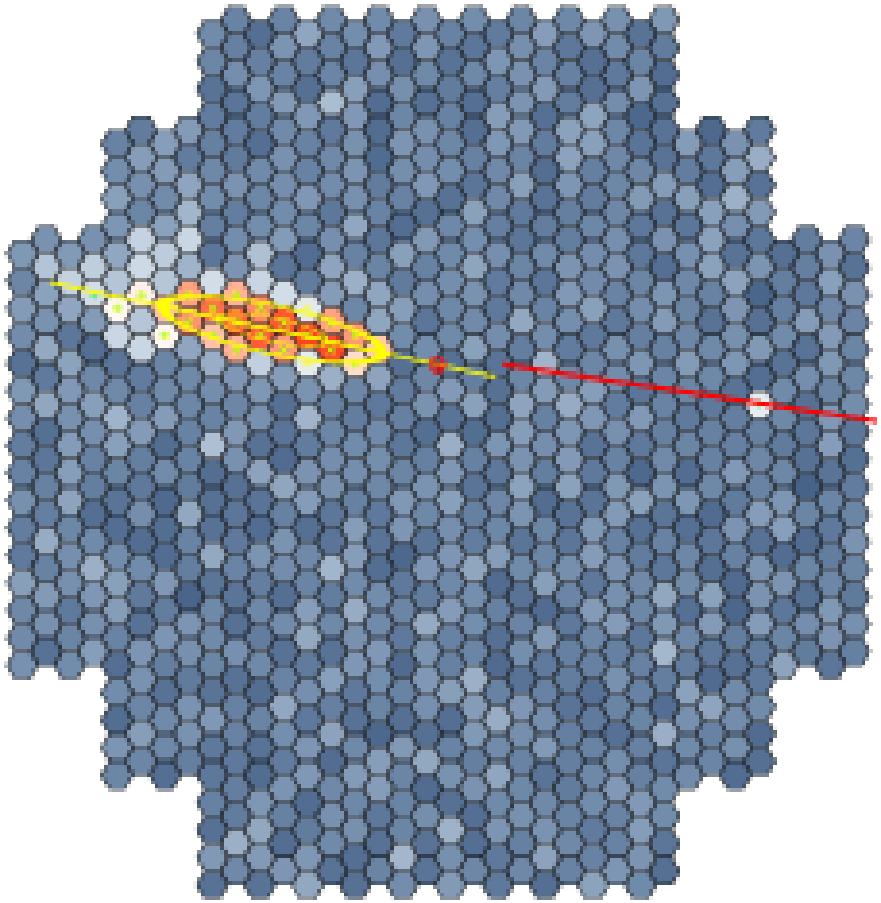




Hillas

Hillas parameters calculation

- Reduction : Sum of all pixels photoelectrons signal per camera
- First momentum : ellipse's position
- Second momentum : ellipse's orientation



Hillas with optimized data format

Optimizations

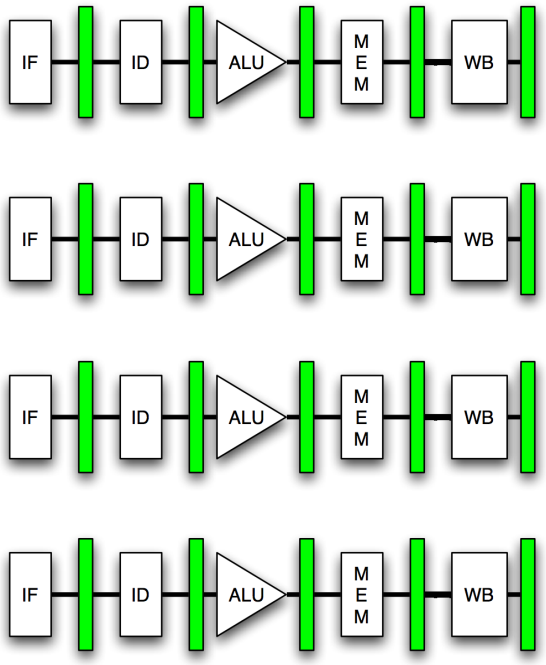
- No ROOT
- Allows CPU data-pre-fetching

Hillas

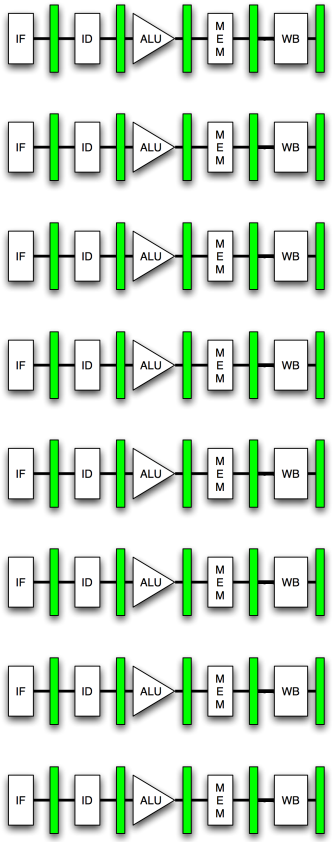
	Speed (cy/el)	Speed up
Initial data format	2125.5	1
Optimized data format	53.1375	40.0

CPU Recent Architectures

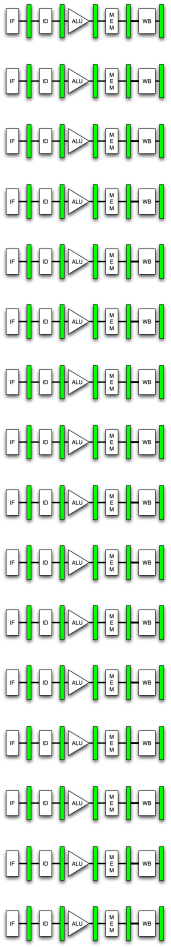
SSE4 4 floats
Instruction set : 2006
CPU : 2007



AVX 8 floats
Instruction set : 2008
CPU : 2011



AVX 512 16 floats
Instruction set : 2013
CPU : 2016



Data format

Efficient only if data are contiguous

Reduction optimization

Reduction Sum of all pixels photoelectrons signal per camera

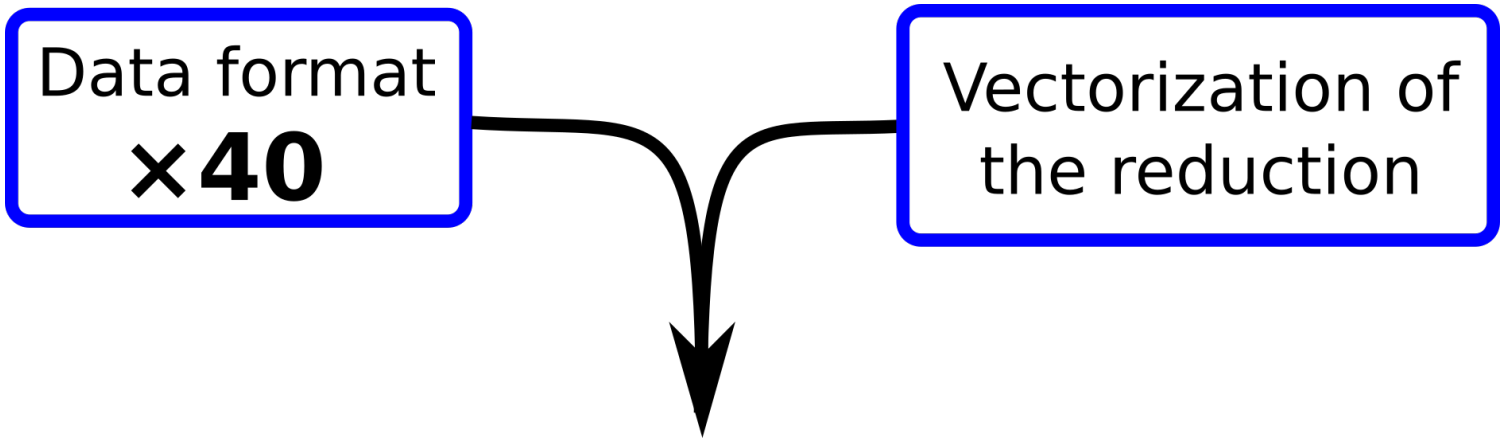
	Speed (cy/el)	Speed up
Classical	2.69842	1
Vectorized (GCC, SSE4)	0.702845	3.8

Reduction optimization

Reduction — Sum of all pixels photoelectrons signal per camera

	Speed (cy/el)	Speed up
Classical	2.69842	1
Vectorized (GCC, SSE4)	0.702845	3.8
Intrinsics Vectorized SSE4	0.226675	11.9
Intrinsics Vectorized AVX	0.11379	23.7

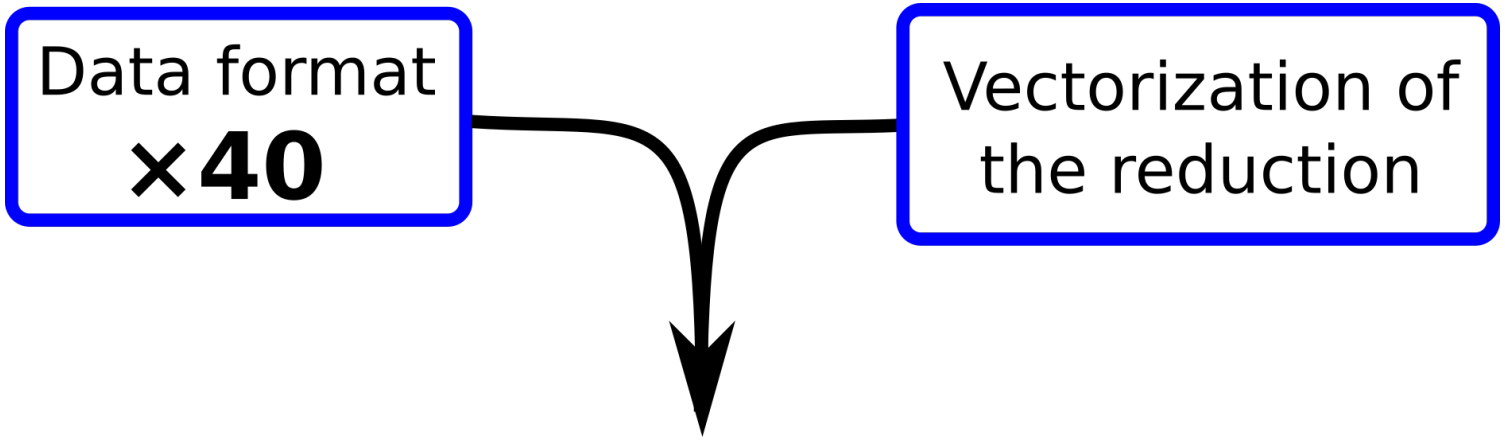
Hillas with optimized format and vectorization



Hillas

	Speed (cy/el)	Speed up
Standard data format	2125.5	1
New data format	53.1375	40.0
+ intrinsics vectorization SSE4	6.39931	332
+ intrinsics vectorization AVX	2.98499	712

Hillas with optimized format and vectorization



Hillas

	Speed (cy/el)	Speed up
Standard data format	2125.5	1
New data format	53.1375	40.0
+ intrinsics vectorization SSE4	6.39931	332
+ intrinsics vectorization AVX	**2.98499**	712

The Hillas parameters calculation costs approximately one reduction (**2.69842 cy/el**)

The more complex is your calculation, the higher speed up you can have

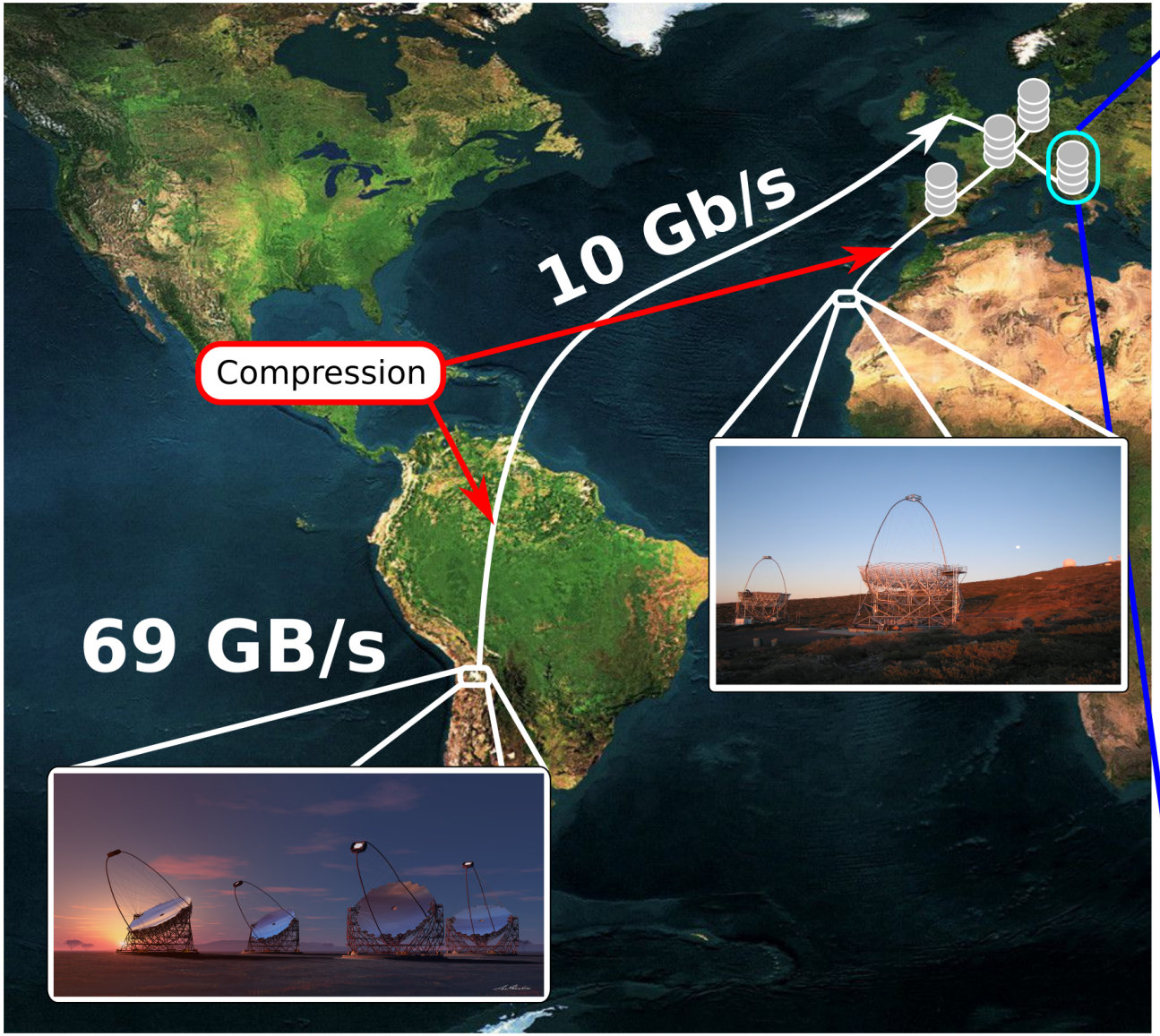
Next analysis optimization

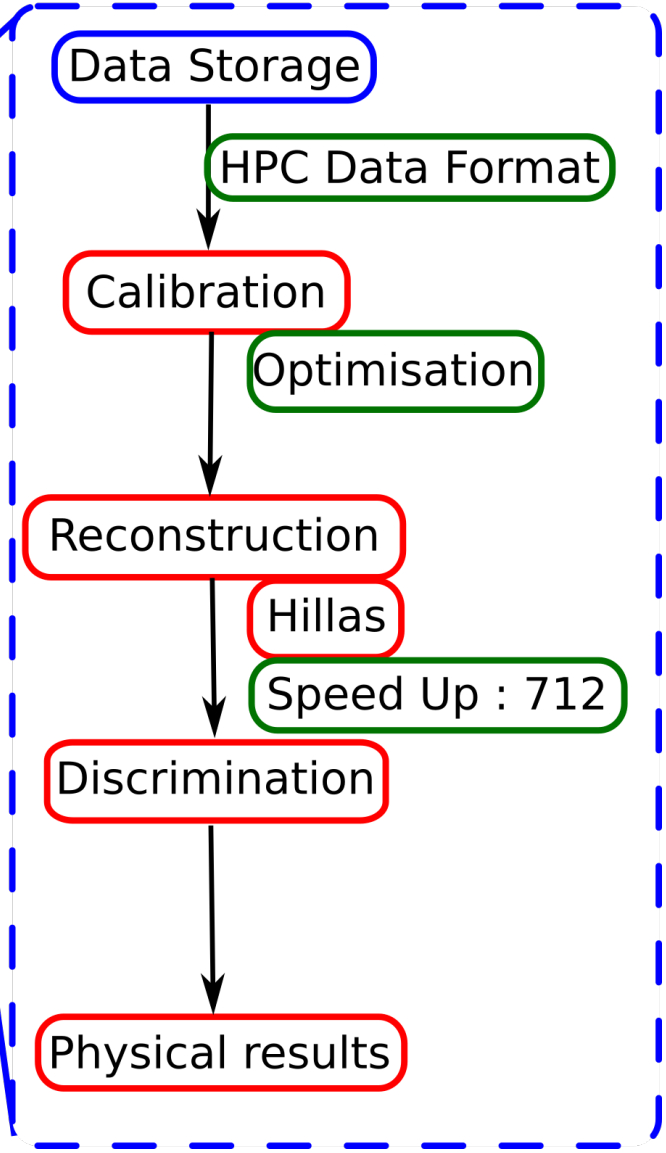
The Hillas method

- Publication soon

The Model++ method

- Simulation of the telescopes
- Images comparison


10 Gb/s
Compression
69 GB/s


Data Storage
HPC Data Format
Calibration
Optimisation
Reconstruction
Hillas
Speed Up : 712
Discrimination
Physical results

Model++ Method

Idea

Modelization of what is observable with the telescopes

- Particle shower simulation (KASCADE, CORSIKA)
- Telescopes answer simulation
- Raw Data compared to the simulated pictures
- Get the physical parameters of the best shower

Model parameters

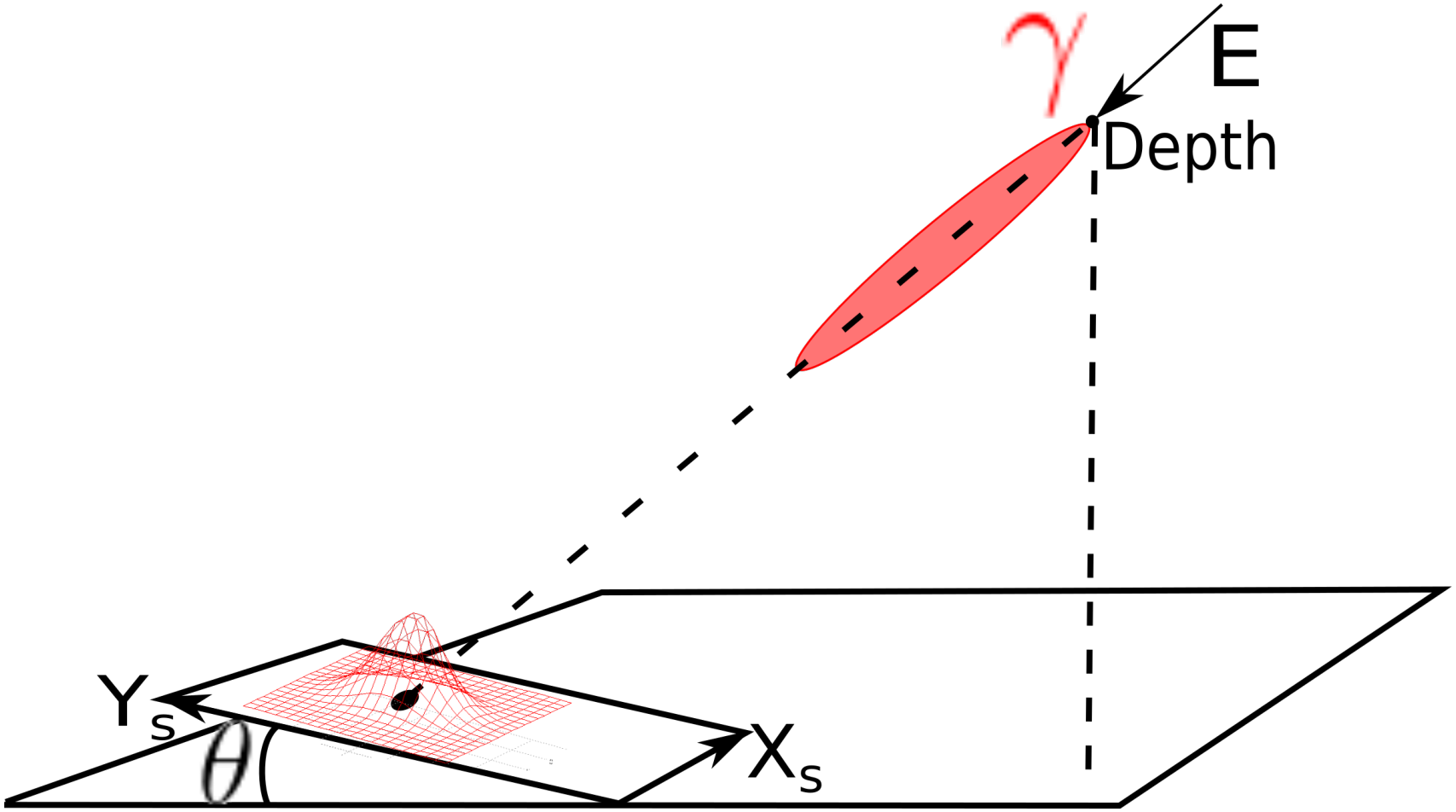
- θ : zenital angle of the telescopes
- E : primary particle energy
- $Depth$: first interaction high
- ρ : distance from the telescope to the shower impact

Adjusting the model

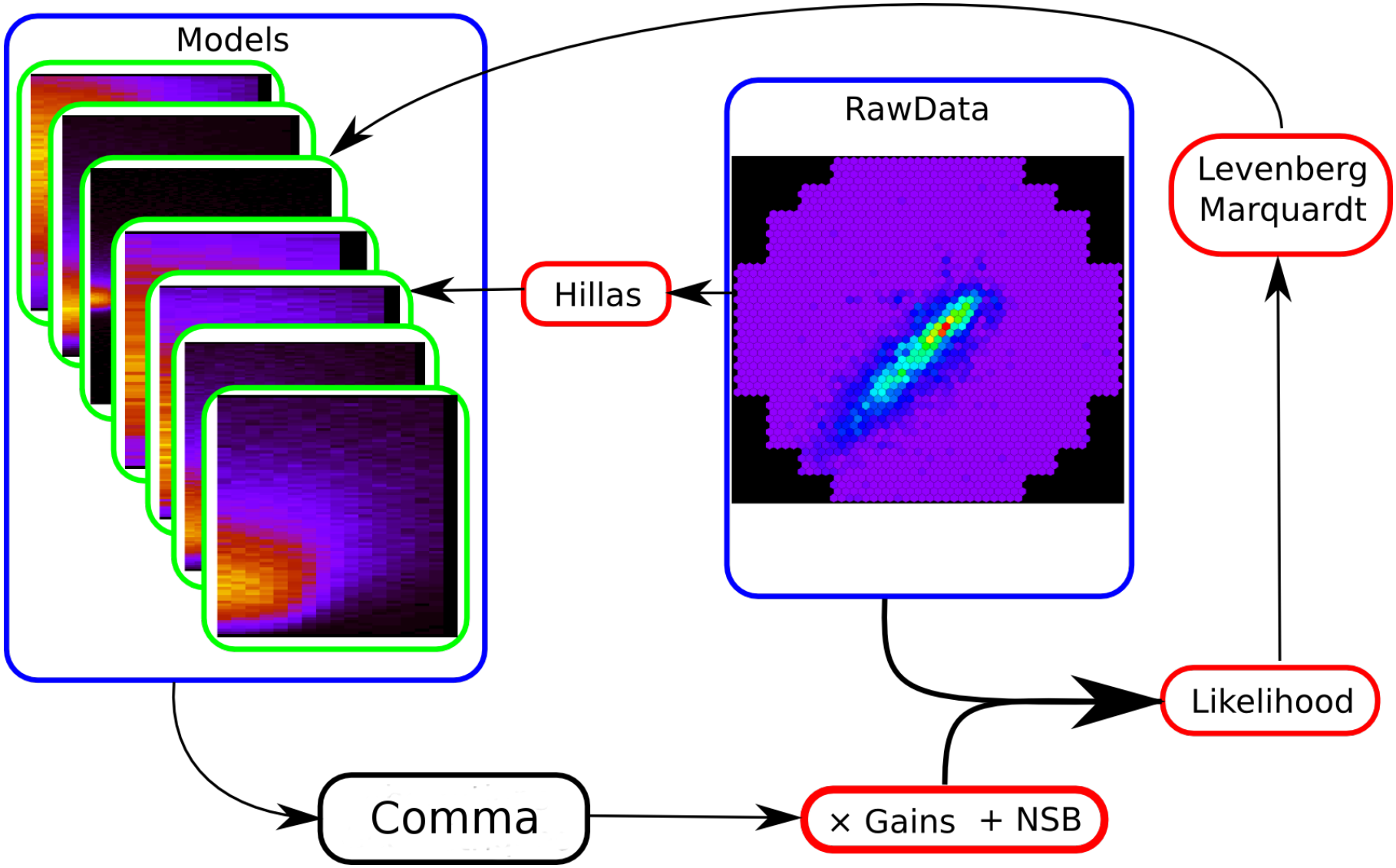
- (x_s, y_s) : model barycentre position in the telescope's camera
- ϕ : model rotation in the telescope's camera

Model++ Method

Model creation



Model++ Algorithm



Model++ Algorithm

Algorithm requiristies

- Stereoscopy (Telescope of the same event in RAM)
- Enought RAM to contain all the telescope's data
- **Telescopes** orderd by **Event**

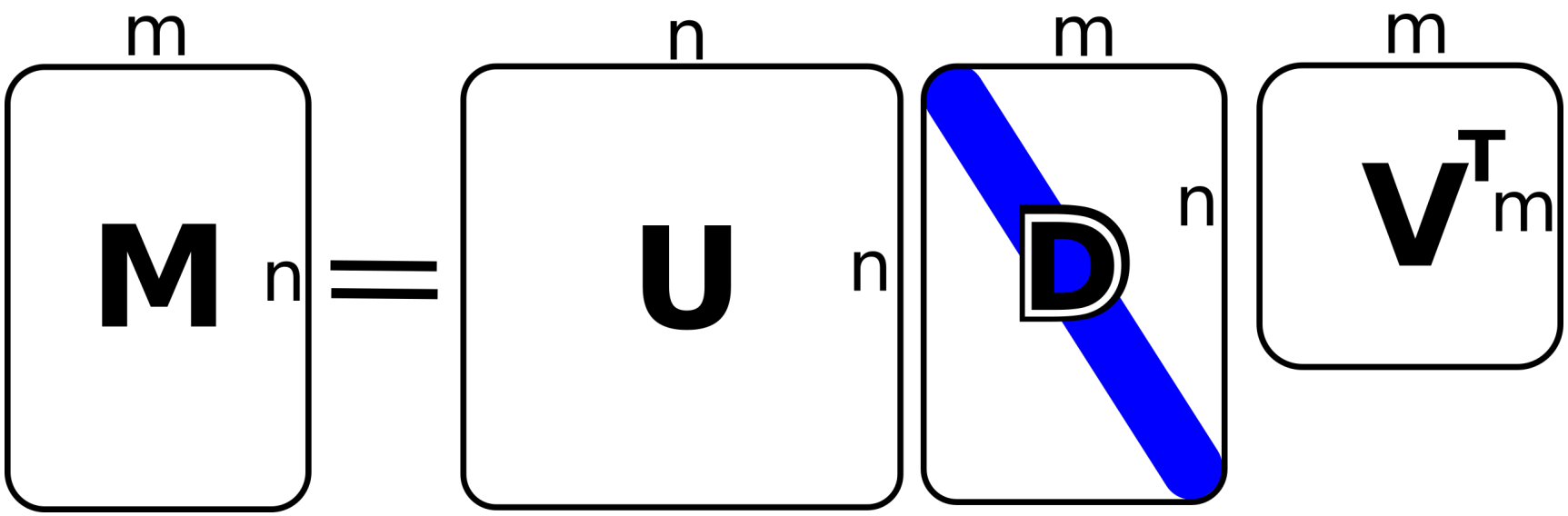
Issues

- Data Scale $\Longrightarrow$ 126 Telescopes, 69 GB/s
  - Too much data for one single computer
    - Lot of communications for the many-core approach
- Data Order :
  - **Event** orderd by **Telescopes** Needs sorting
    - Sort a huge data volume $\Longrightarrow$ Huge RAM Data access time unrealistic

The SVD method

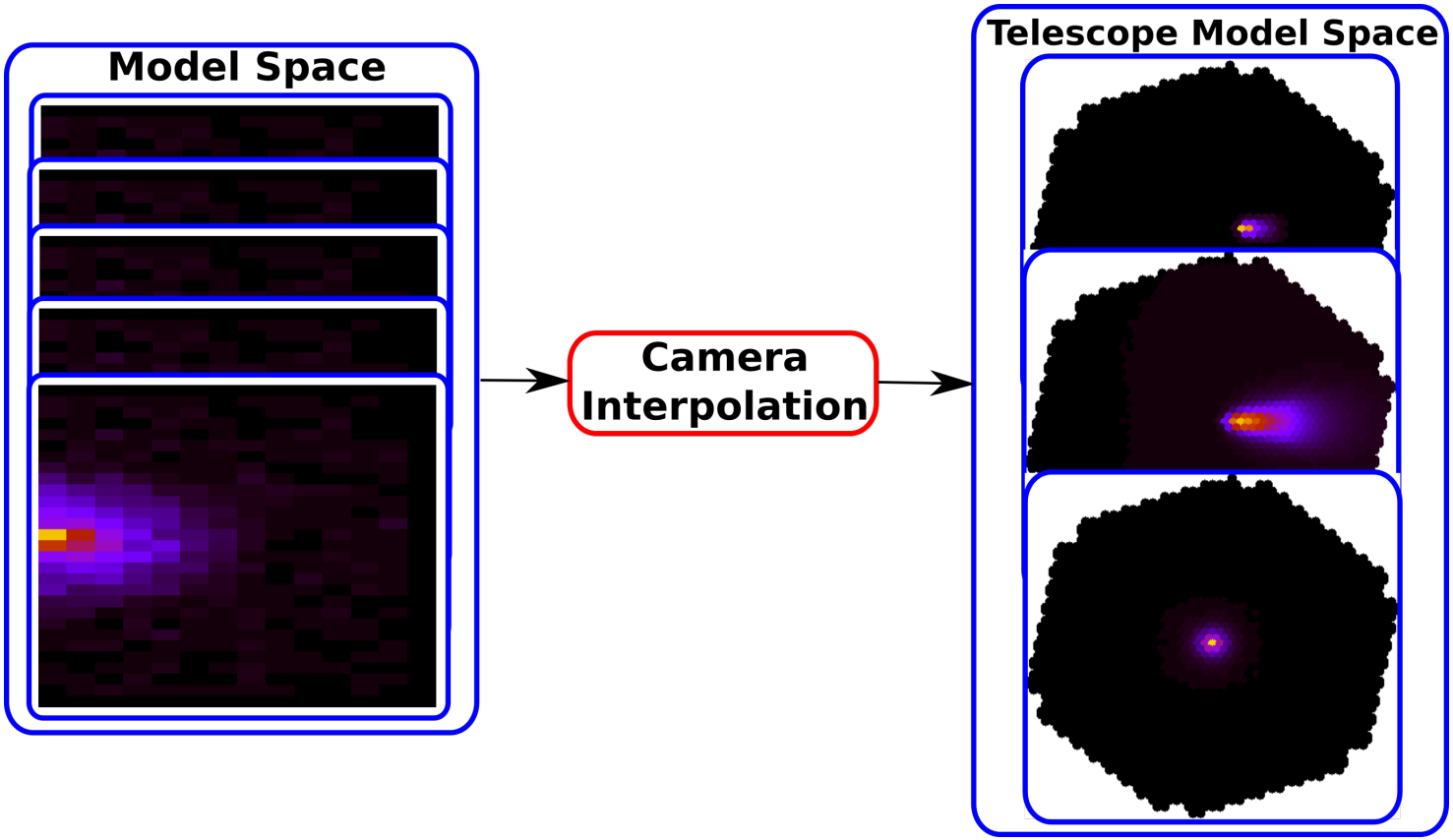
The Single Value Decomposition

$$M = UDV^T$$



The SVD method

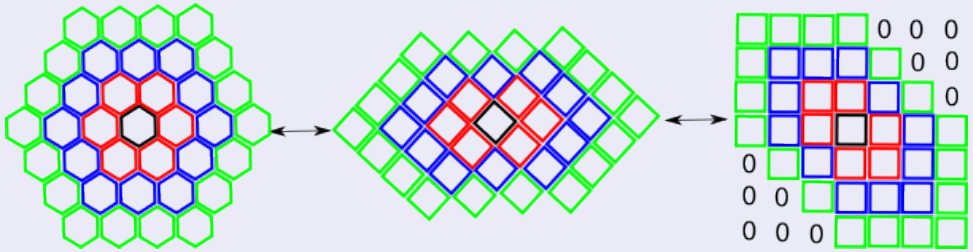
Model conversion : Dragon

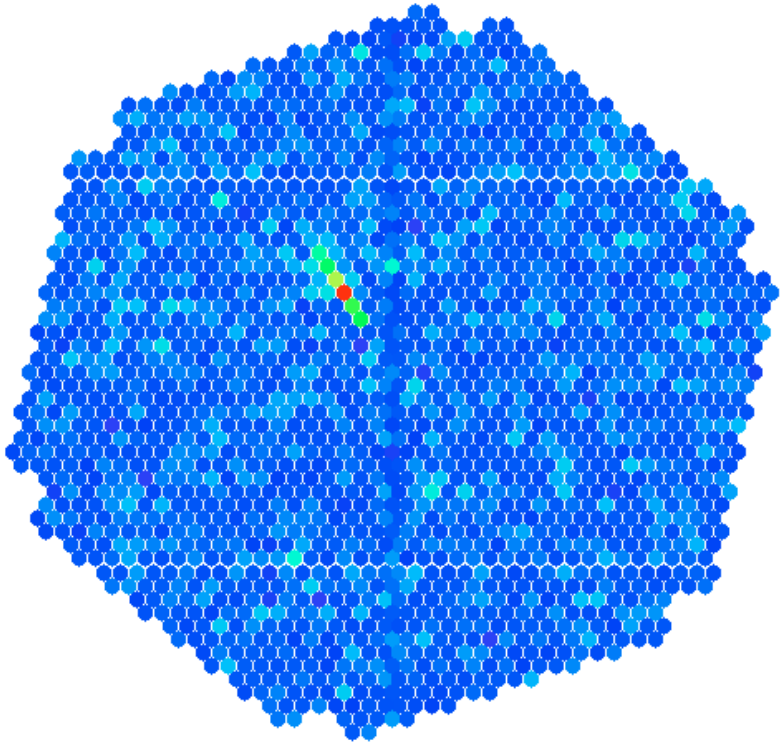


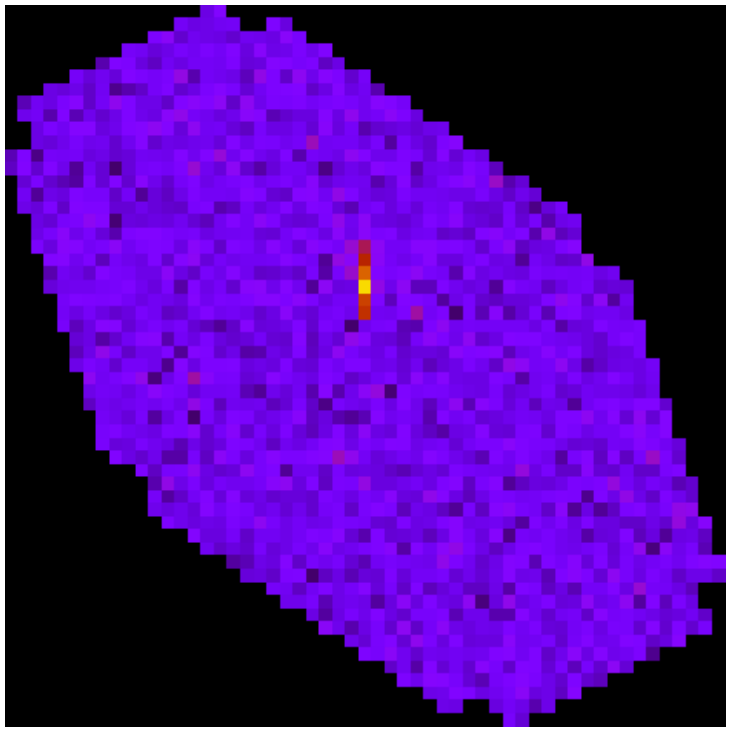
The SVD method

Dragon camera to matrix

Camera to matrix conversion

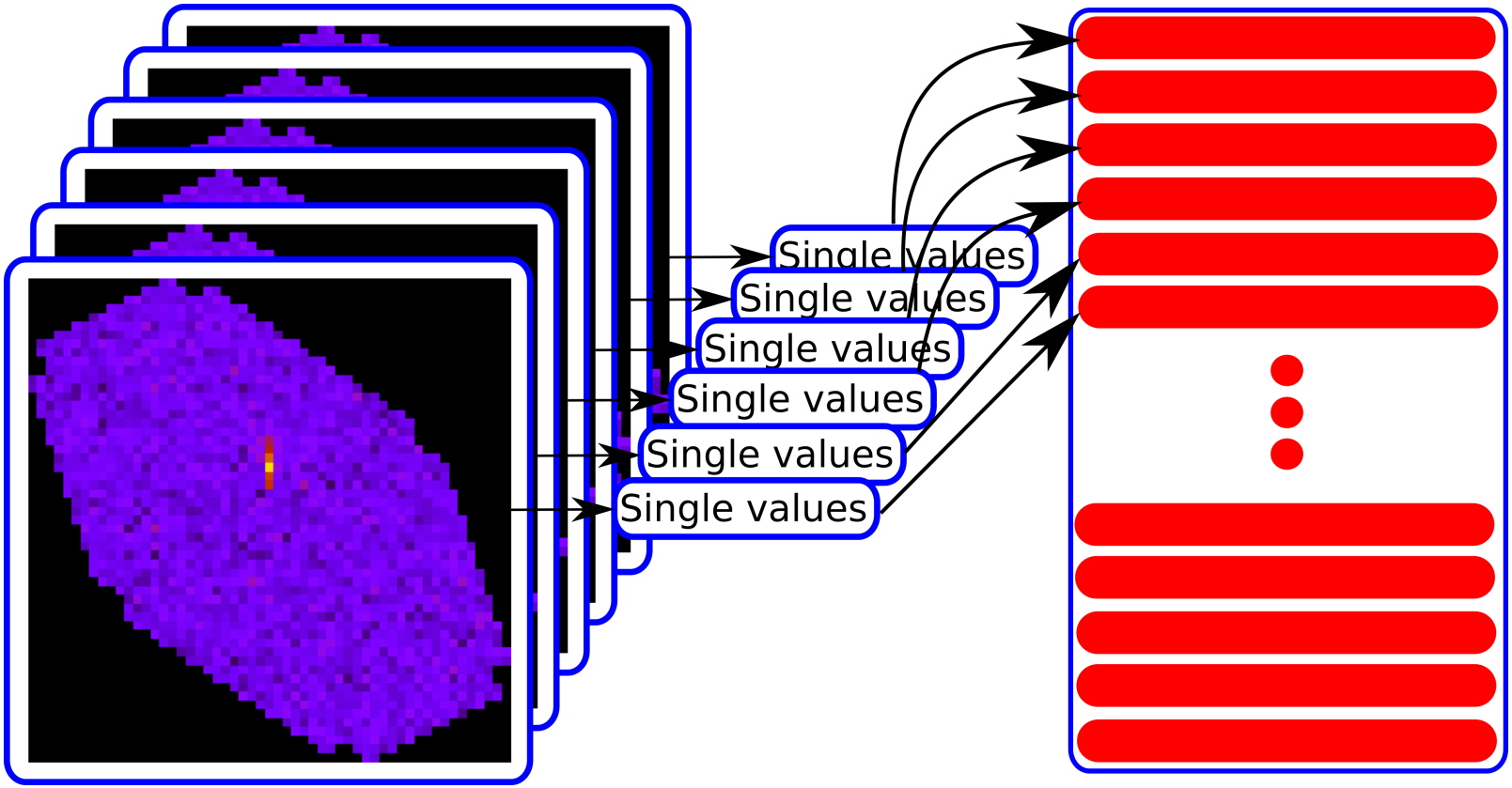






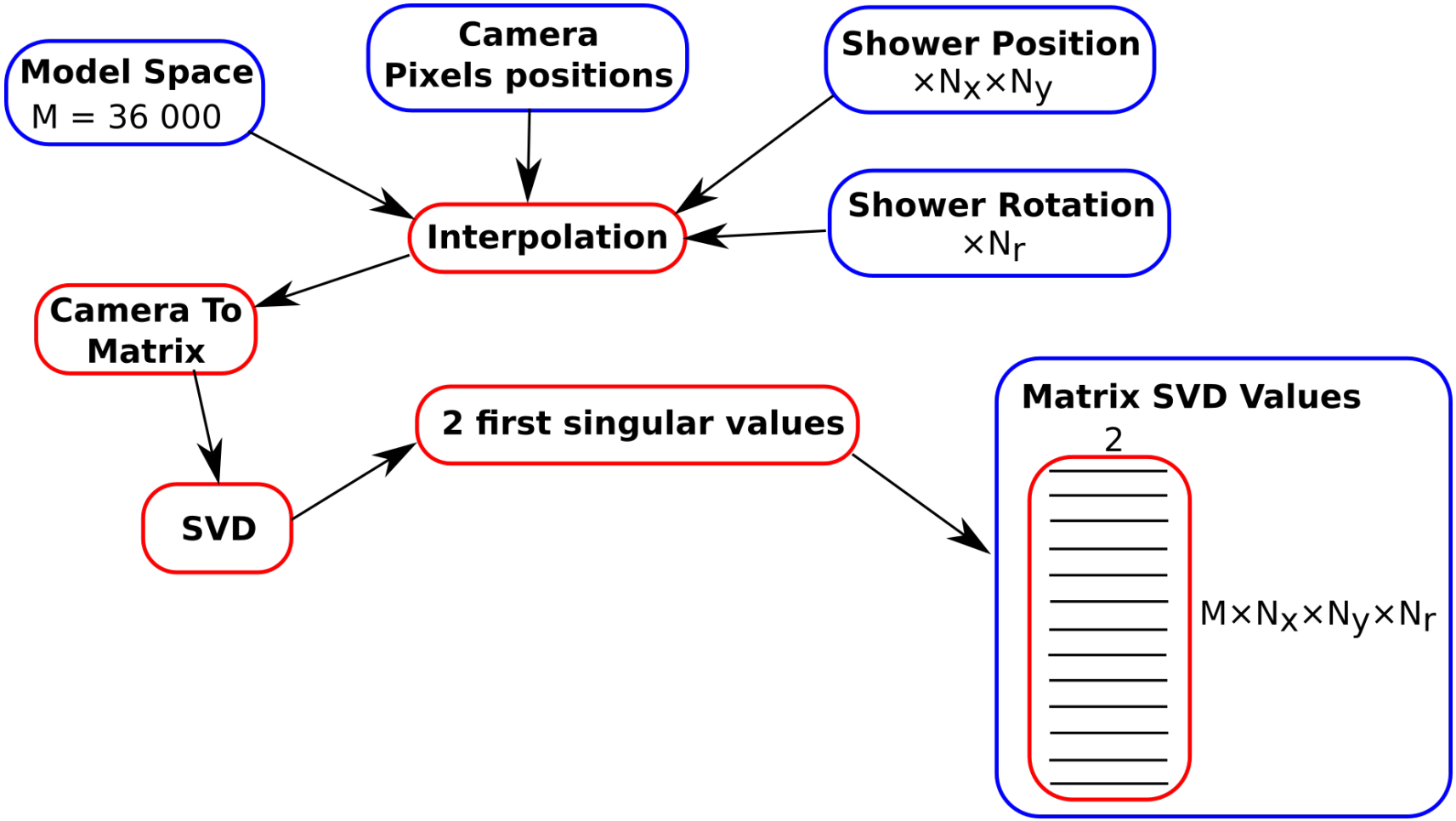
The SVD method

SVD on camera : Dragon



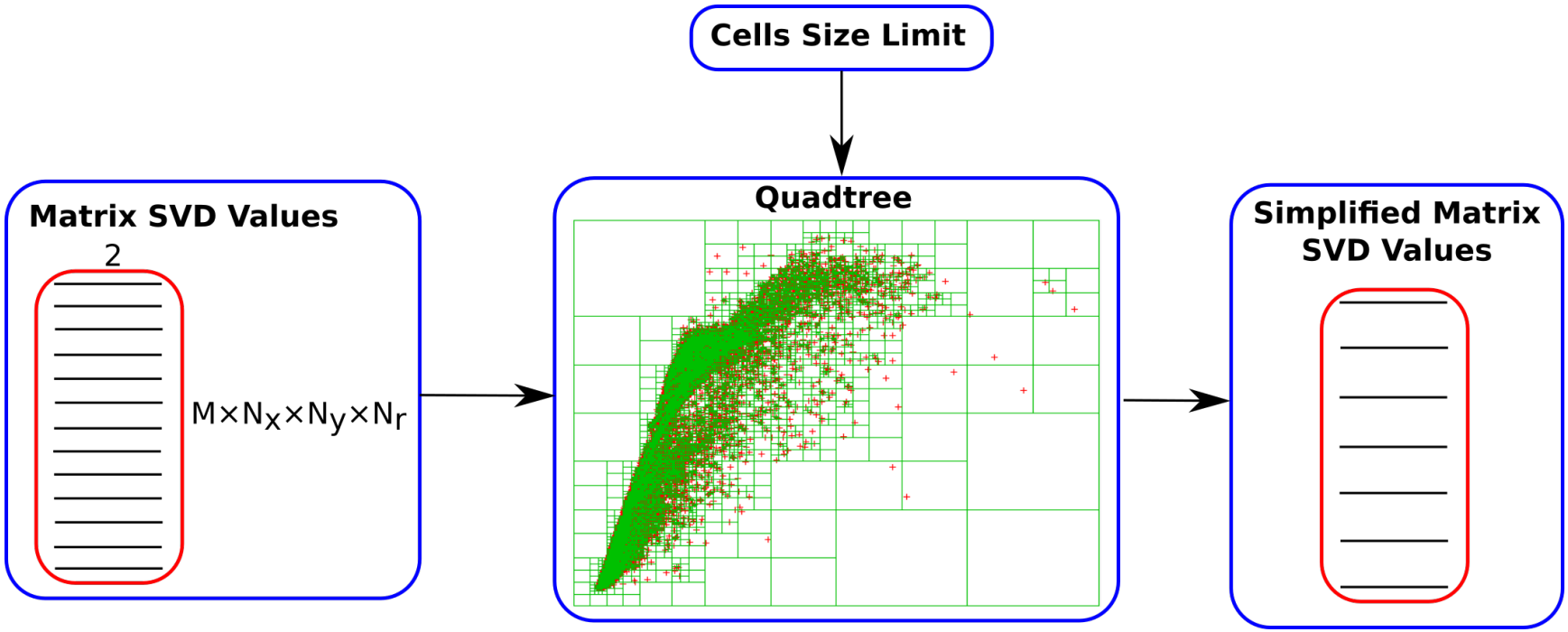
The SVD method

From Model Space to SVD matrix



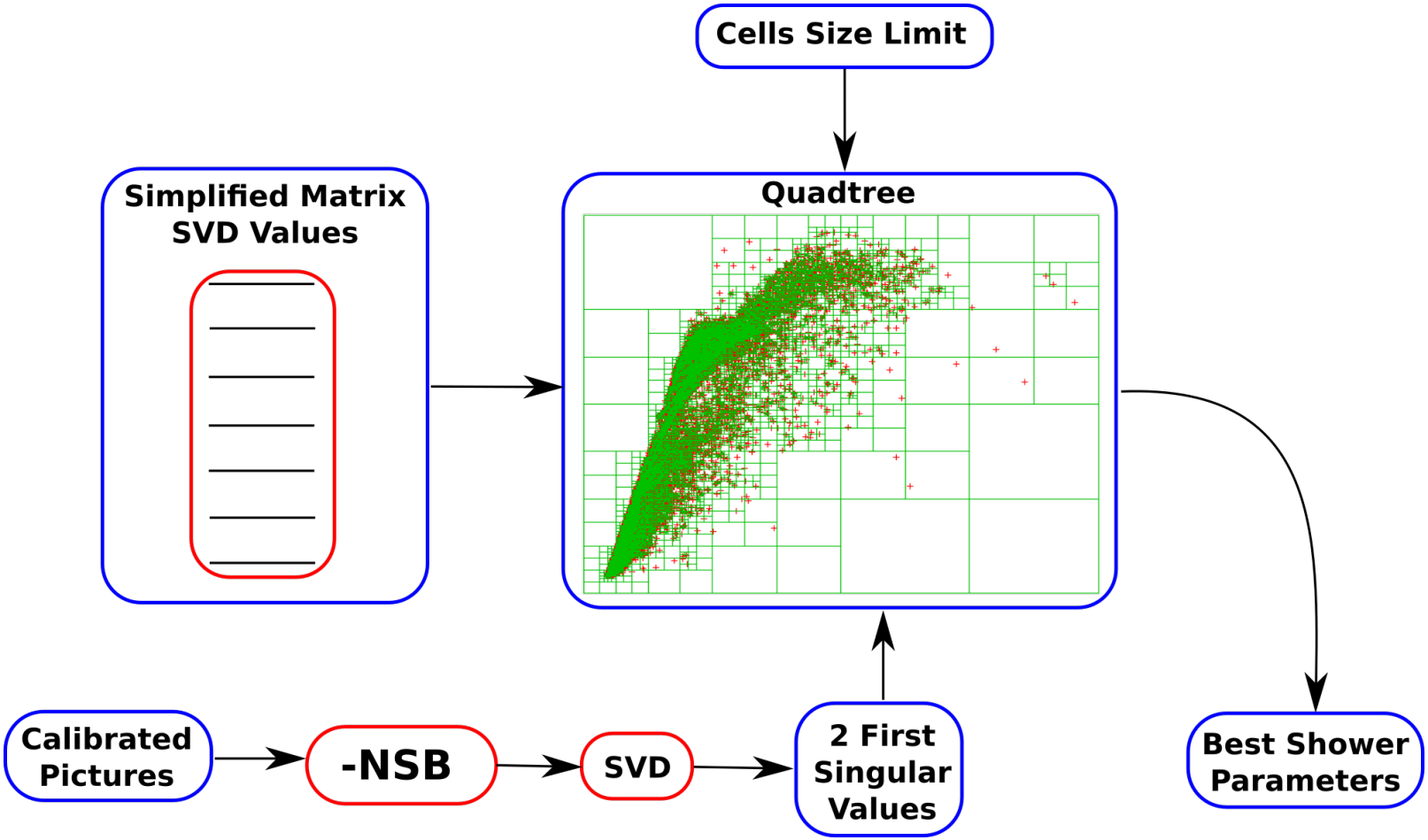
The SVD method

SVD matrix simplification



The SVD method

SVD Reconstruction



Performances of several Cores/CPUs/Nodes

Parallelization

- Load-balancing
- Optimization of CPU use

Official ctapipe Parallelization system

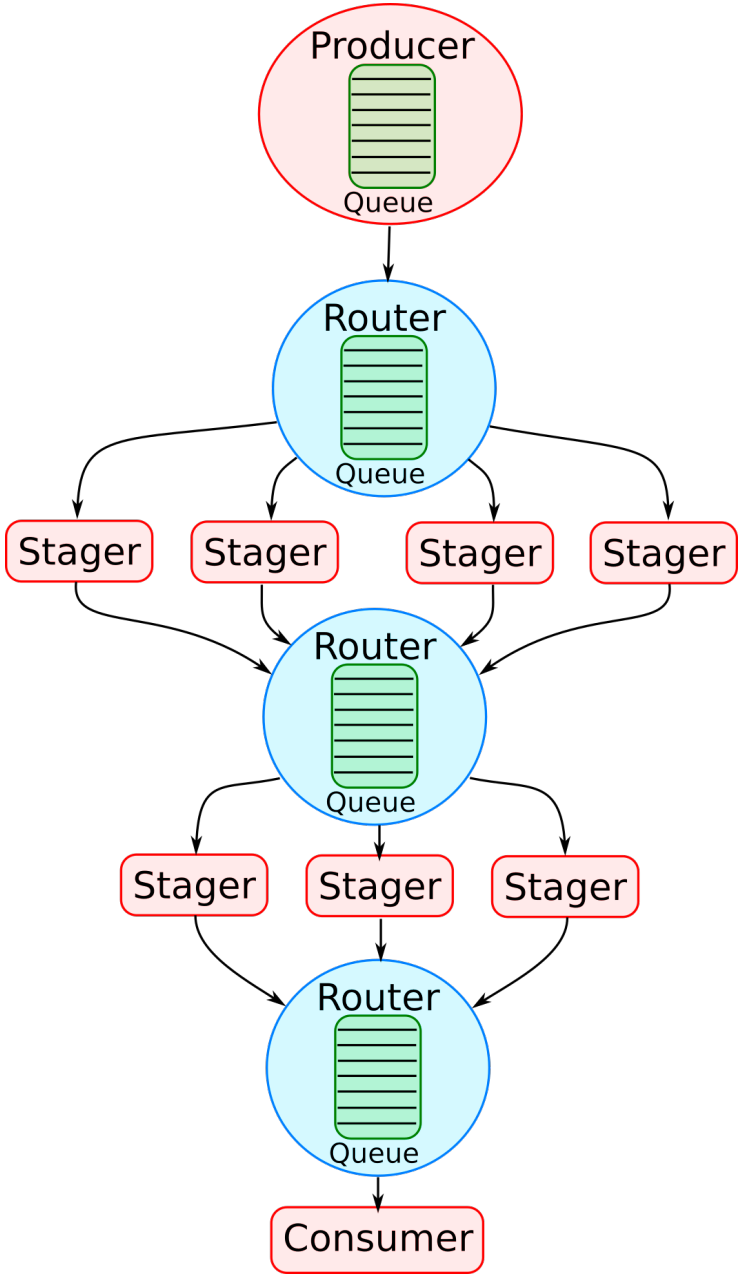
Functionnalities

- Flexible
- Chains the analysis steps
- Parallelizes the analysis steps
- Load-balances the analysis steps

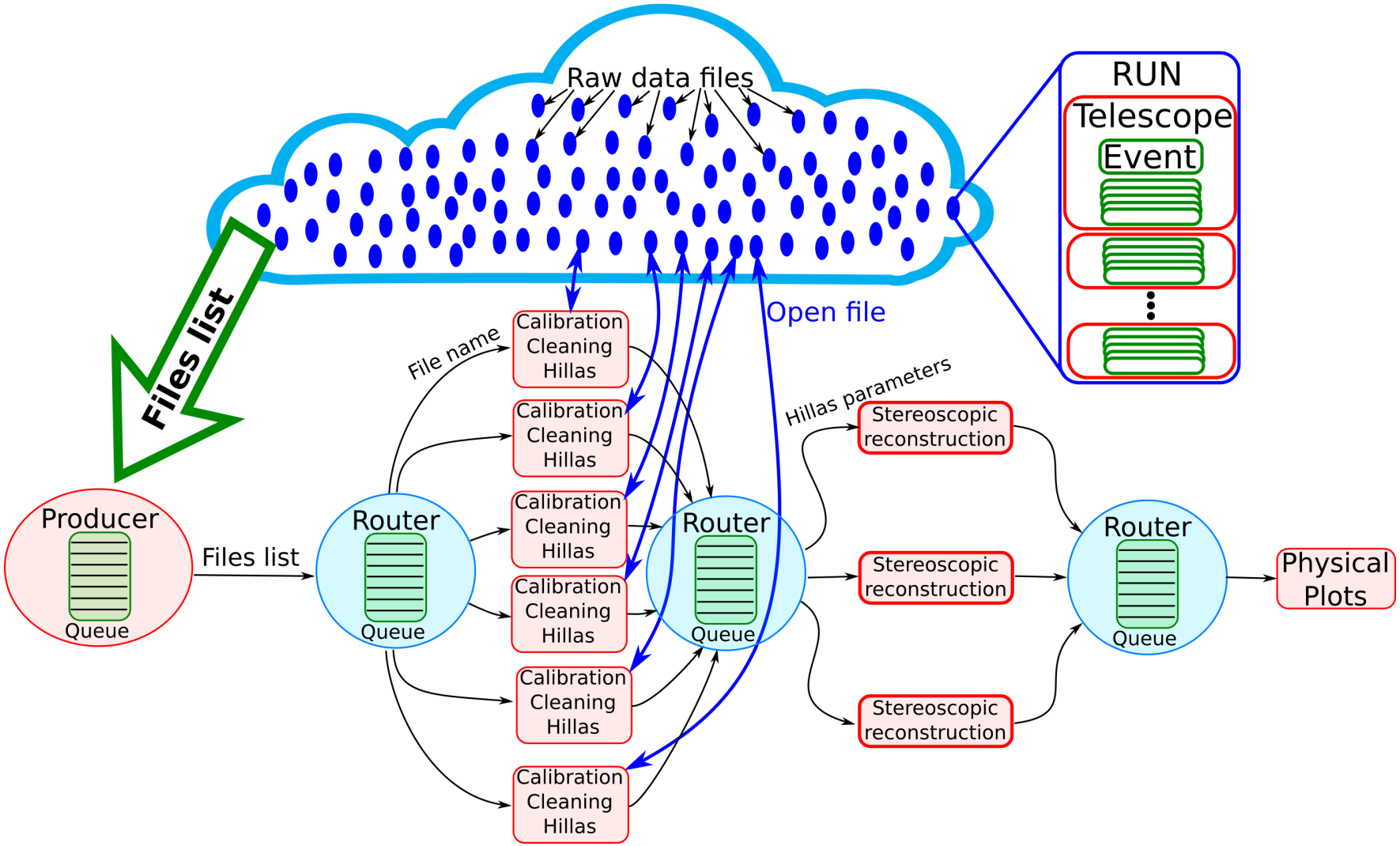
ZeroMQ

- Communication between stager and router
- Independent of core/node/computer
- No All-to-all communications

Jean Jacquemier



CTA pipe for analysis



Software

Implementation

- C, C++, Python (wrapper)
- Intrinsics (vectorization by hand)
- Loop unrolling
- Loop interleaving

Framework

- Open source
- Licence CeCILL-C
- https://gitlab.in2p3.fr/CTA-LAPP/PLIBS_8

Conclusion

Data transfert (69 GB/s $\Longrightarrow$ 10 Gb/s)

- Data Data compression : Pulication ongoing

Data Analysis (69 GB/s)

- Optimized data format
- Vectorization
  - Hillas parameters calculation (speed up 712, publication soon)
- Other algorithms
  - Model++ optimization ongoing
  - SVD Model optimization ongoing

Backups

CPU Architecture

CPU Architecture

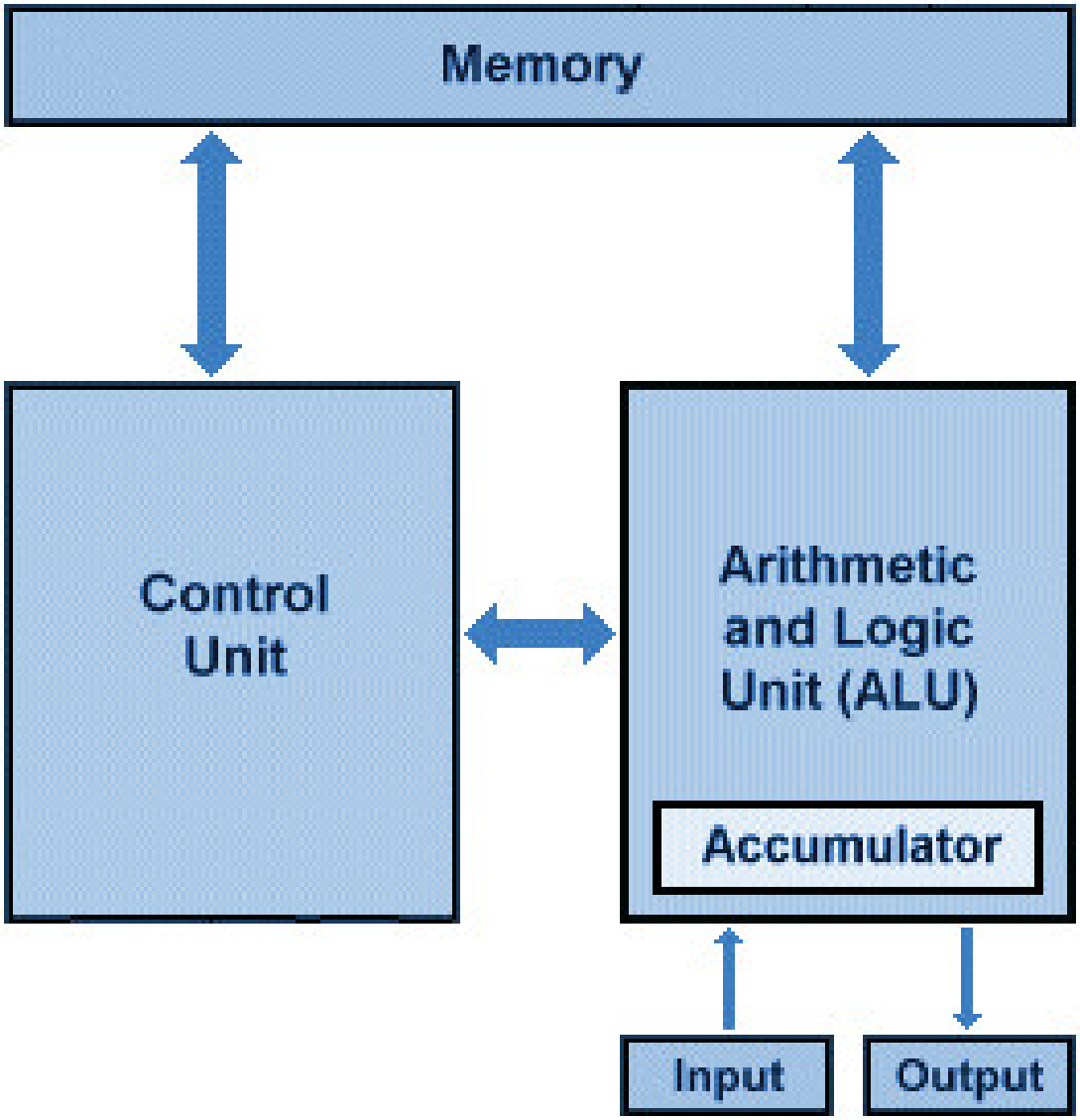
Von Neumann architecture 1945

Definition

Cycle : basis unit of time in a CPU

Time

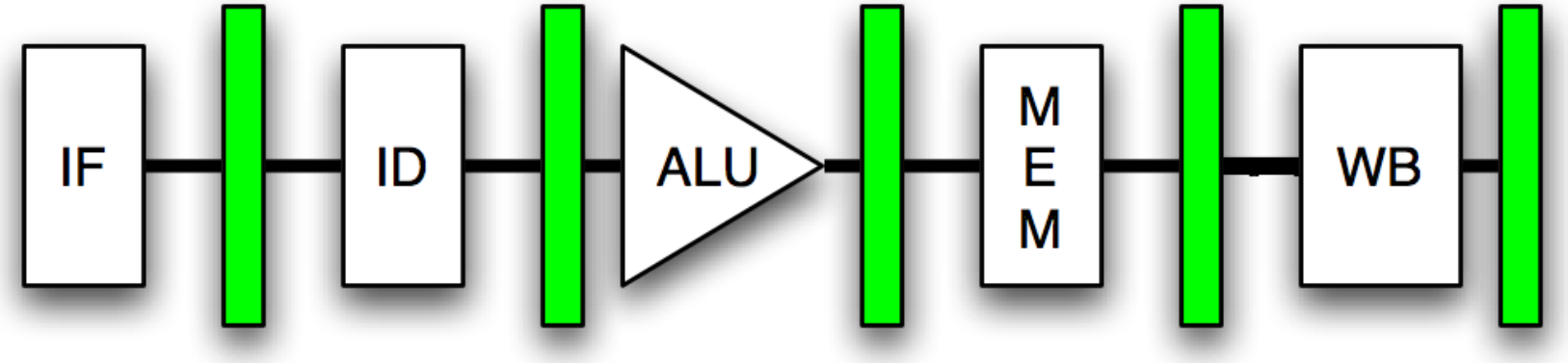
- 1 cycle per elementary operation (load, store, add, ...)
- 4 cycles per whole operation ($c = a + b$)



CPU Architecture

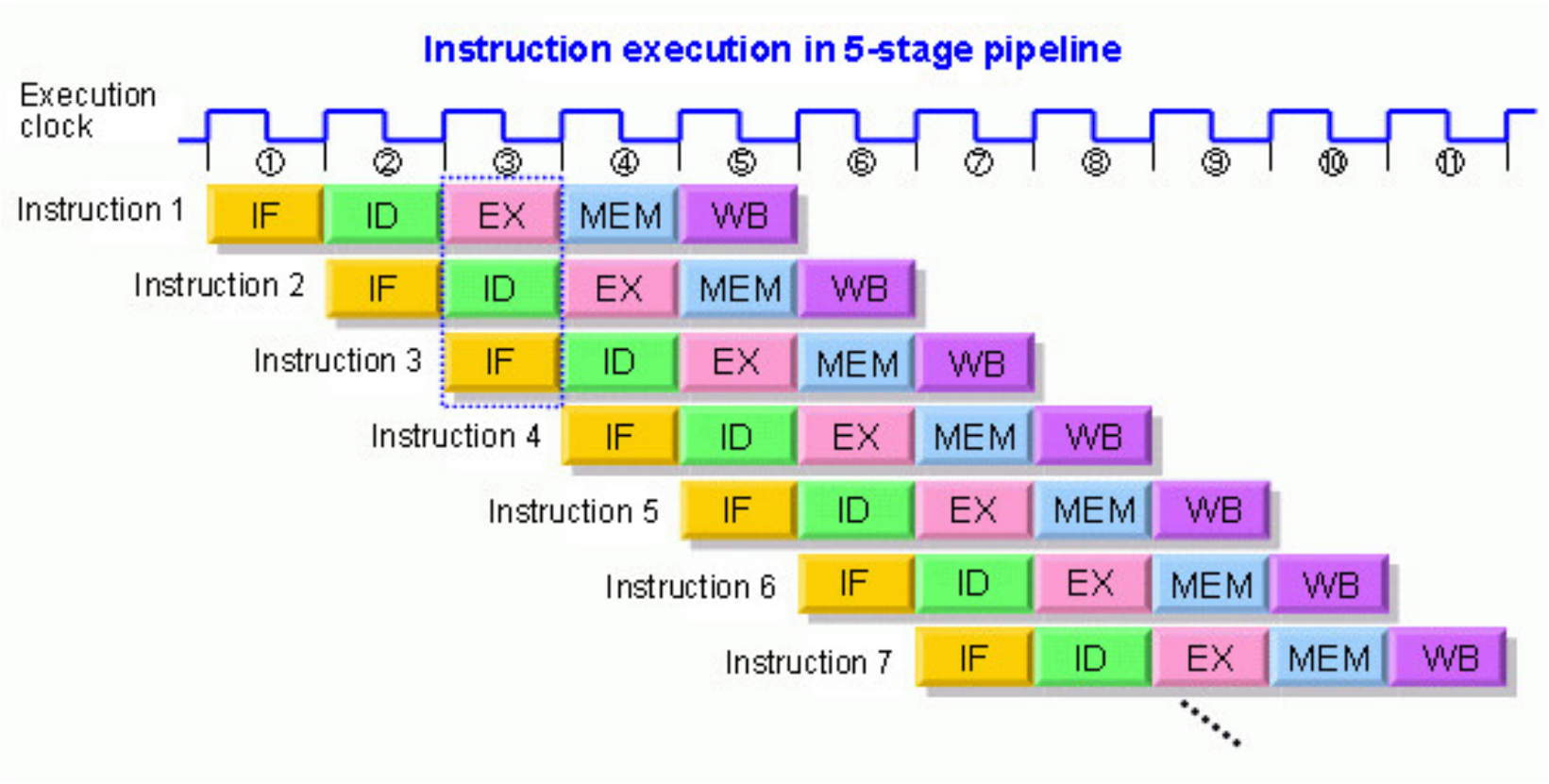
Pipeline approach

- IF : Instruction Fetch
- ID : Instruction Decode
- ALU : Execution
- MEM : Memory
- WB : Write Bytes



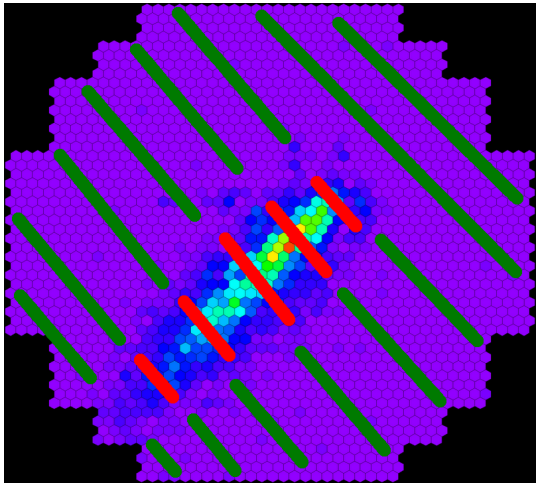
CPU Architecture evolution

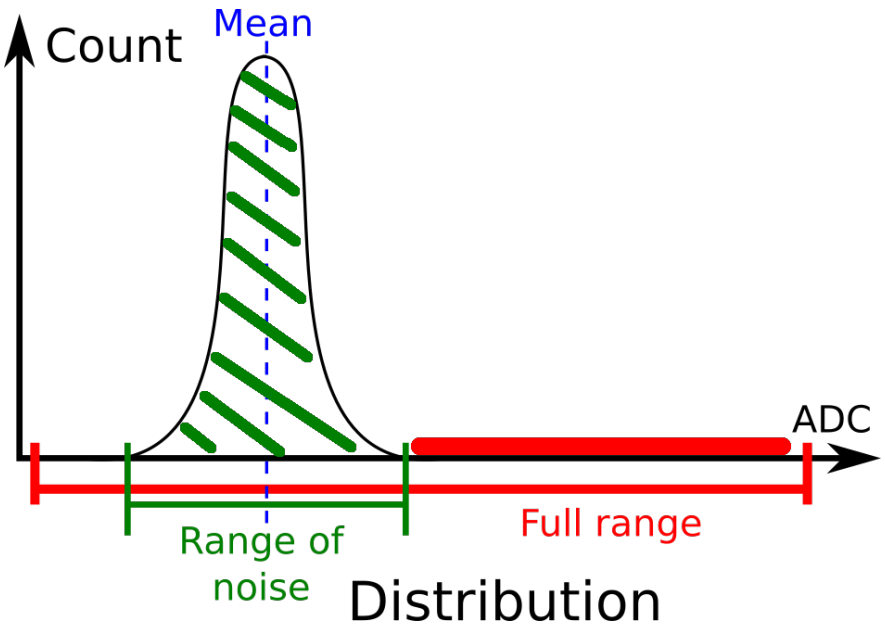
Pipeline using



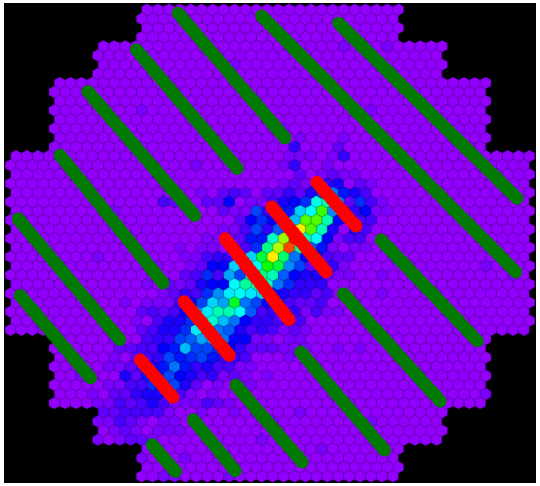
Data lossless compression

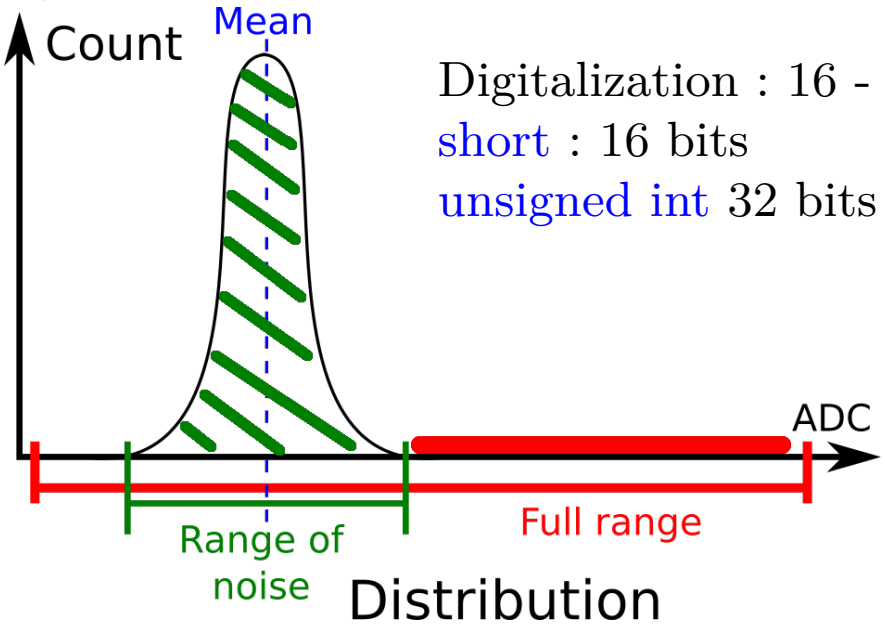
Digitalization : 16 - 24 bits
short : 16 bits
unsigned int 32 bits



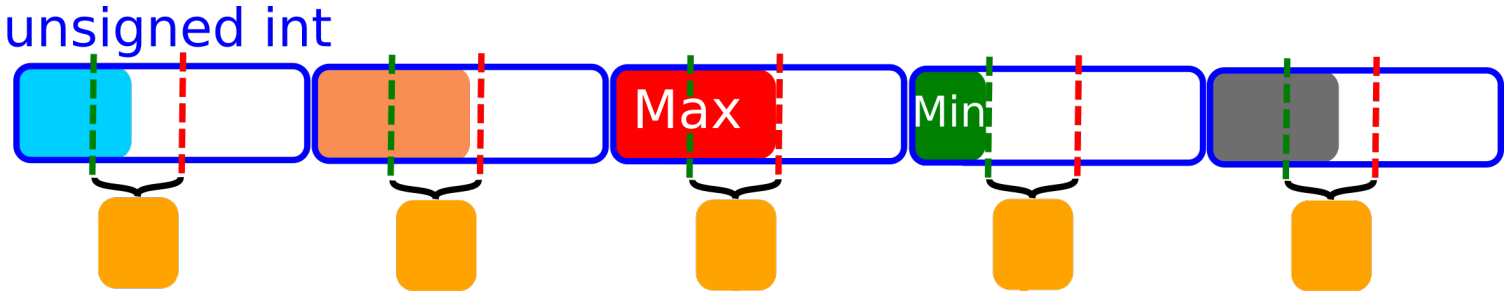


Data lossless compression

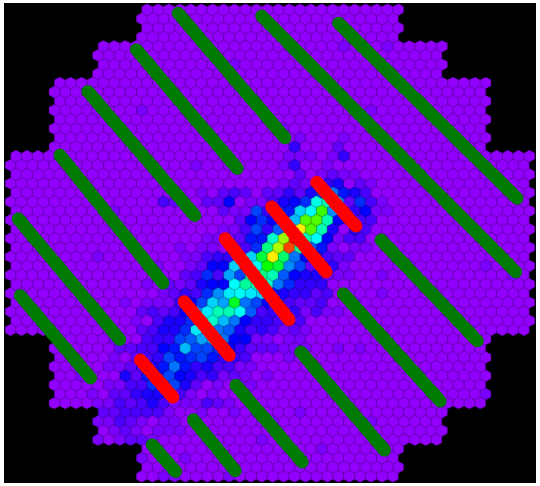


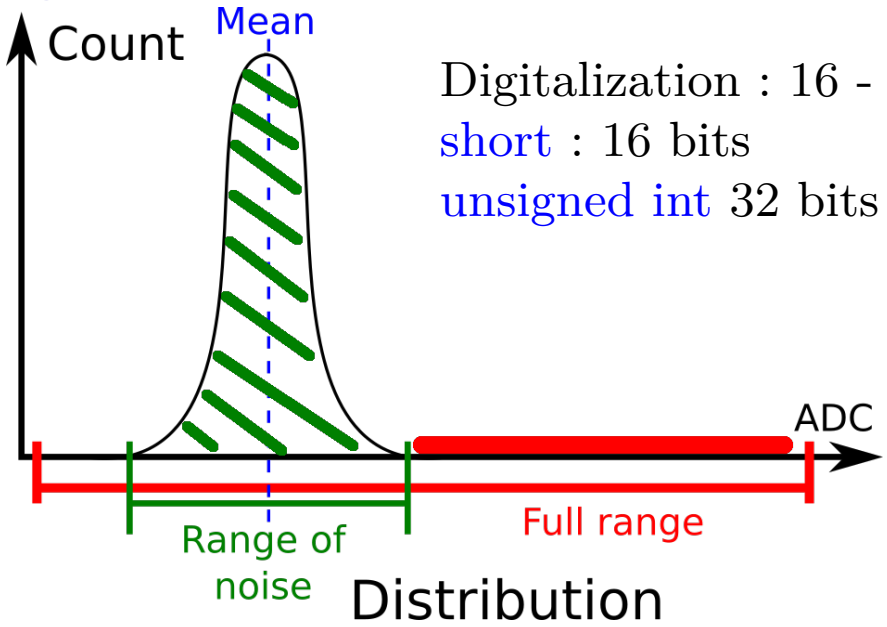


Digitalization : 16 - 24 bits
short : 16 bits
unsigned int 32 bits

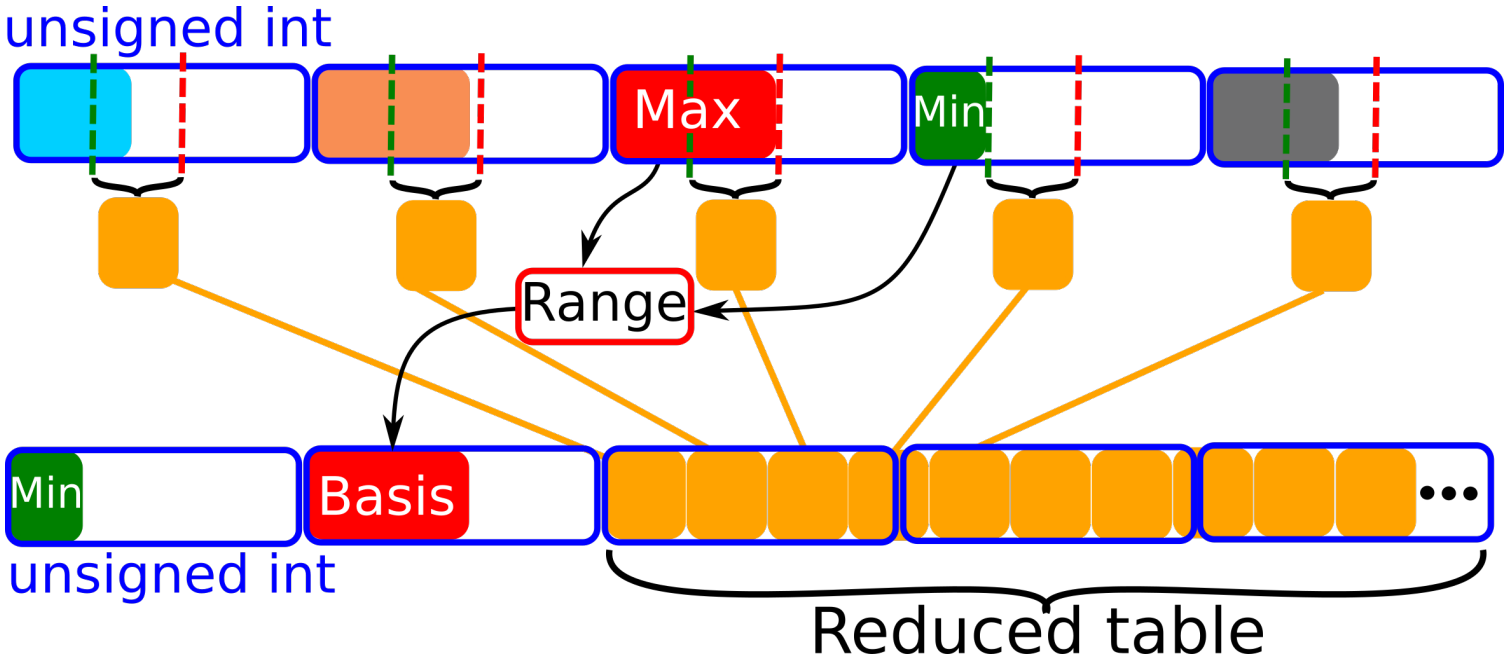


Data lossless compression



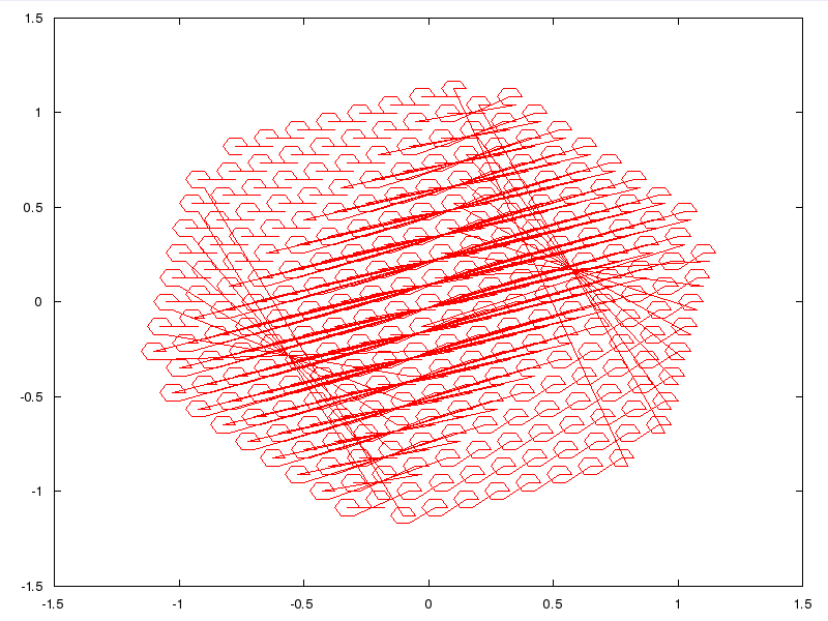


Digitalization : 16 - 24 bits
short : 16 bits
unsigned int 32 bits

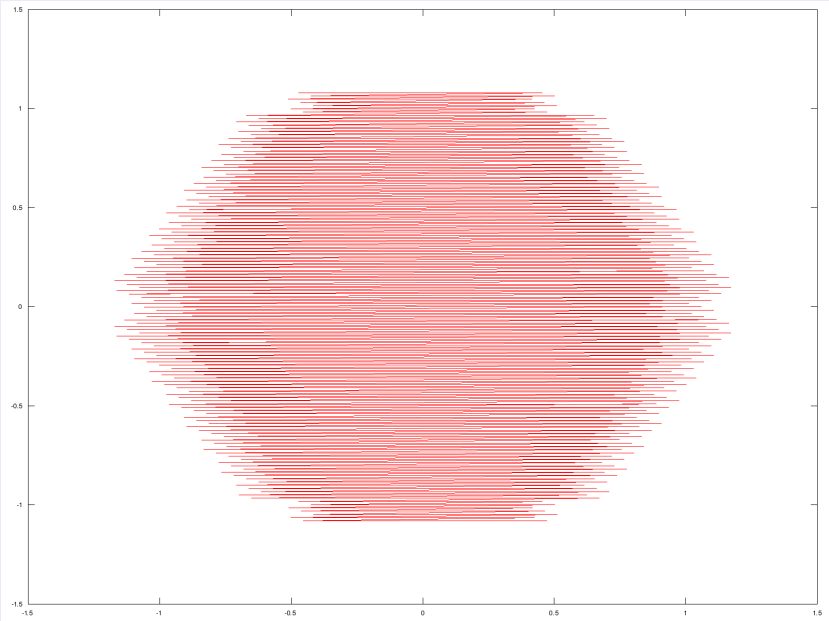


Sorting the camera's pixel to guaranty data locality

Before pixels sorting



After pixels sorting



Advantages

- Guaranty data locality
- Allows fast pixel's neighbours search
  - Faster Hillas cleaning
- Helps the data lossless compression