Solving the *CMS High Granularity Calorimeter's* Data Processing Challenges with *Heterogeneous Computing*

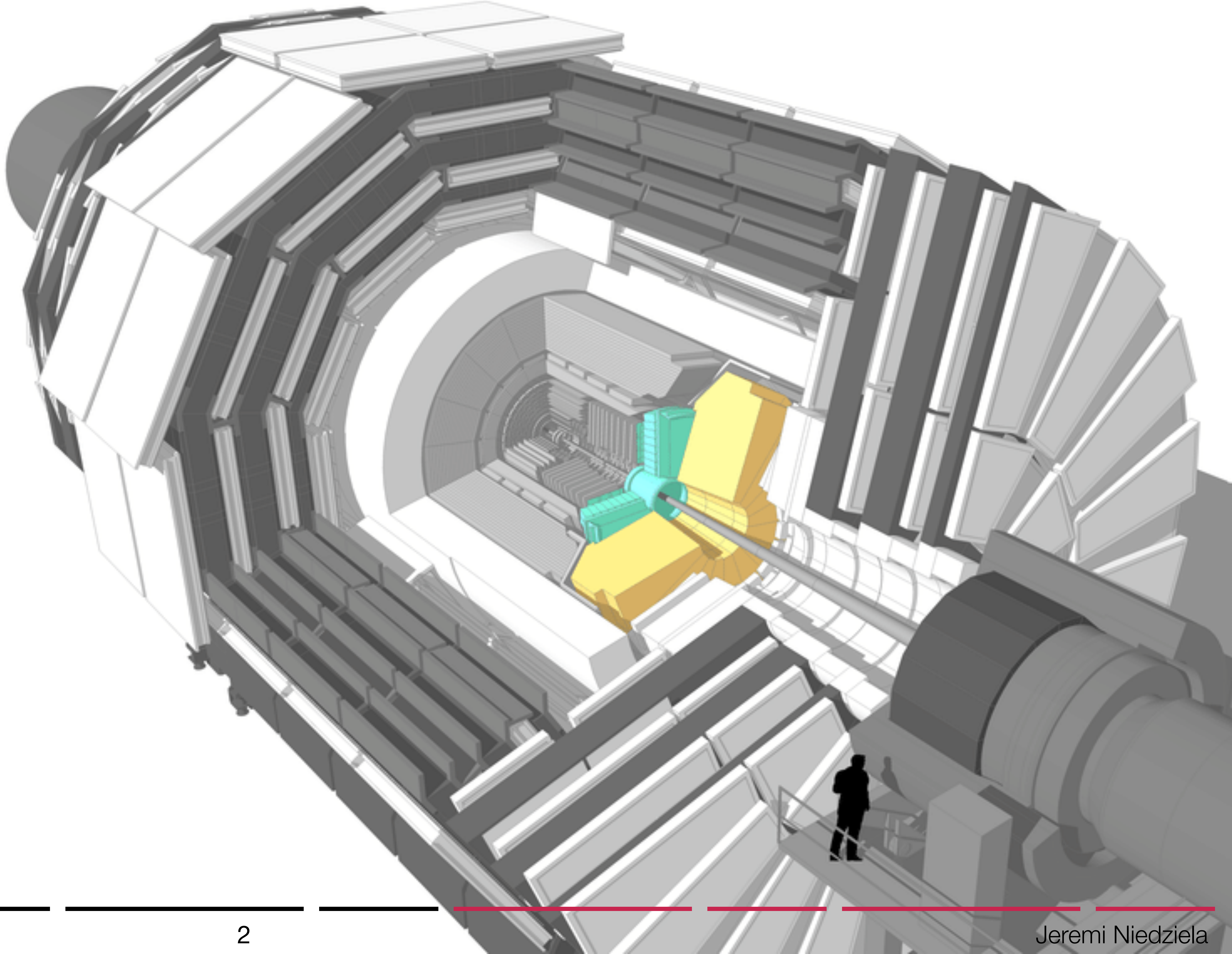
FH Scientific Computing Platform
DESY, 1/07/2024

Juliette Alimena, Freya Blekman, **Jeremi Niedziela**, Lovisa Rygaard

High Granularity Calorimeter

What's HGCal?

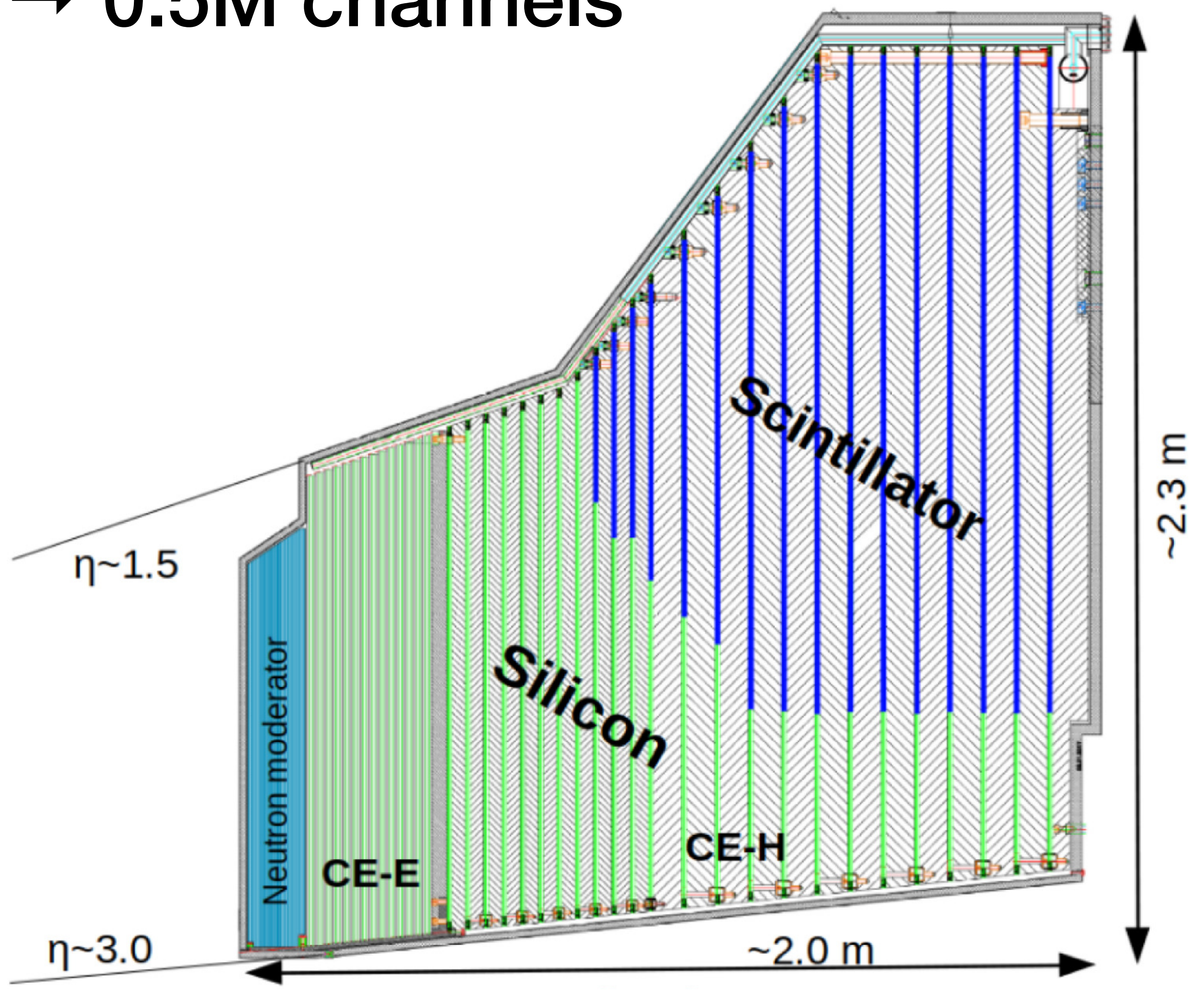
- **new detector for CMS Upgrade**
  → starting in Run 4 (≈2029)
- replacing ECAL & HCAL

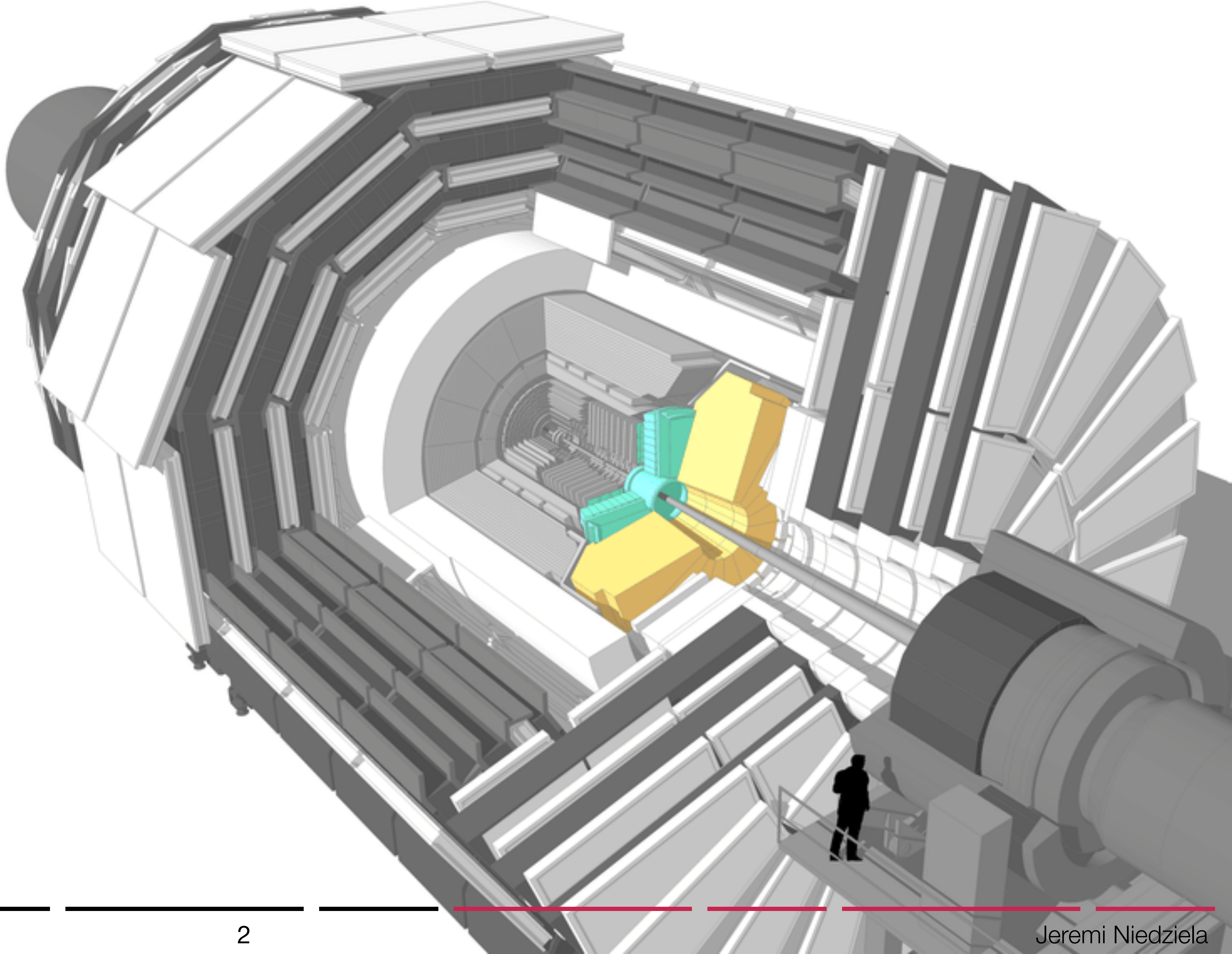


High Granularity Calorimeter

What's HGCal?

- **new detector for CMS Upgrade**
  → starting in Run 4 (≈2029)
- replacing ECAL & HCAL
- a **tracking calorimeter**
  → **47 layers**, $1.5 < \eta < 3.0$
- CE-E (EM part): silicon sensors
  → **6M channels**
- CE-H (Hadronic part): silicon + scintillator tiles
  → **0.5M channels**





HGCal Challenges

Challenges

- **6.5M channels** → HGCal event size ≈40% of the total CMS size
- need to **decode data fast** to fit the strict High Level Trigger (HLT) time requirements

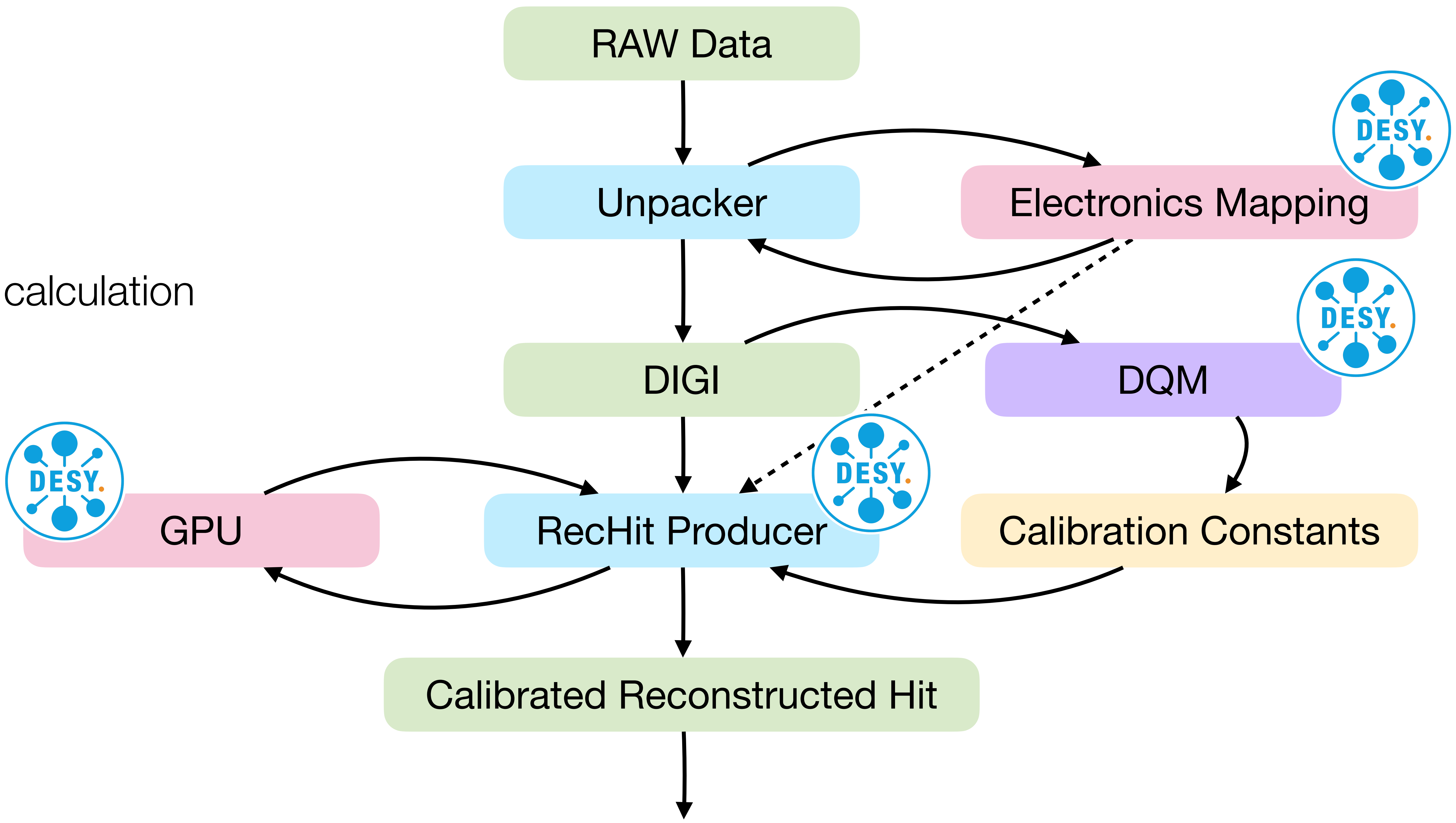
HGCal Challenges

Challenges

- **6.5M channels** → HGCal event size ≈40% of the total CMS size
- need to **decode data fast** to fit the strict High Level Trigger (HLT) time requirements

Reconstruction flow

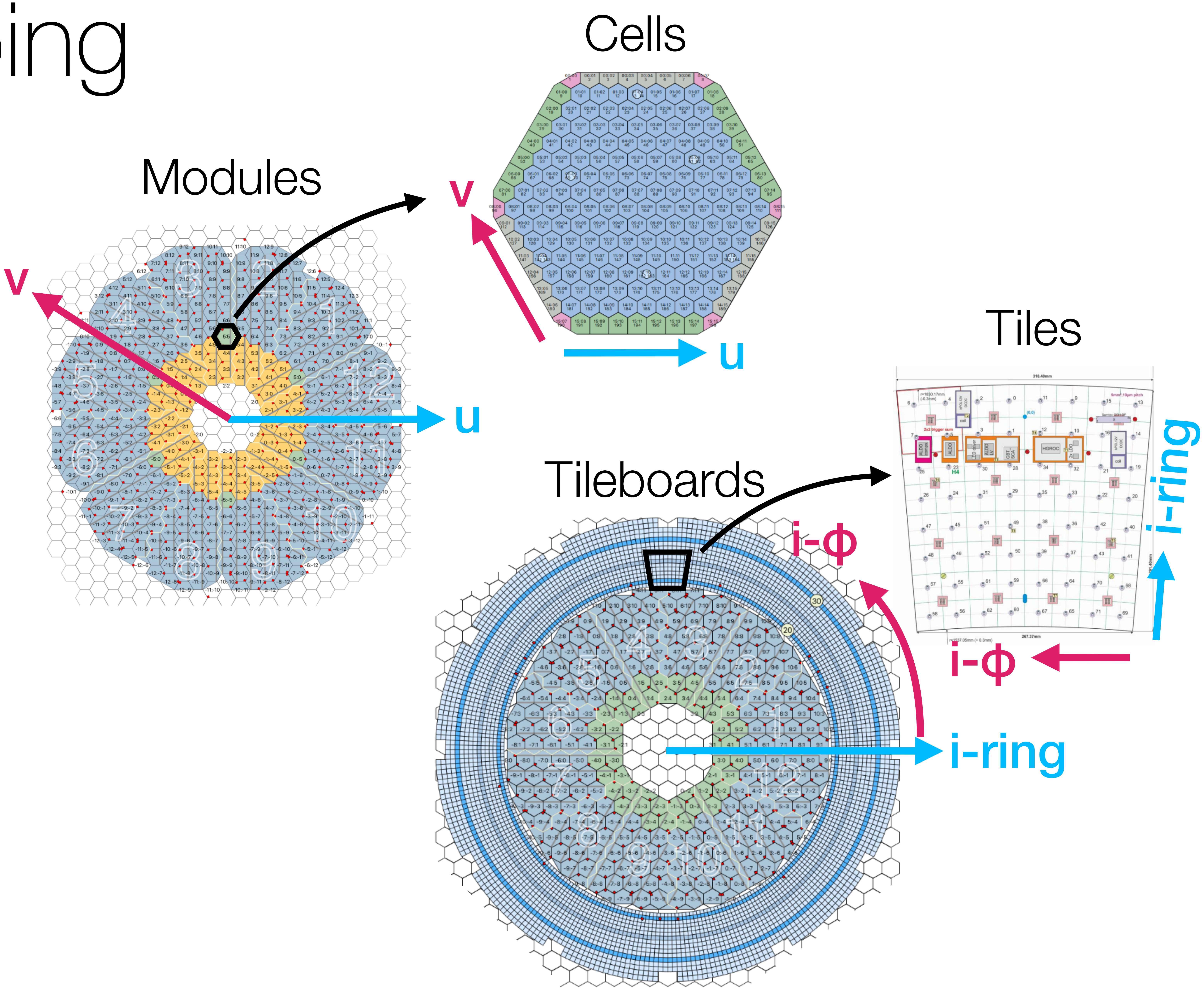
- the chain involves:
  - unpacking+**mapping**
  - Data Quality Monitoring with calibration constants calculation
  - RecHit production and **calibration**
  - then passing on to the reconstruction…
- **DESY** is involved in multiple steps in this process



Electronics Mapping

Mapping

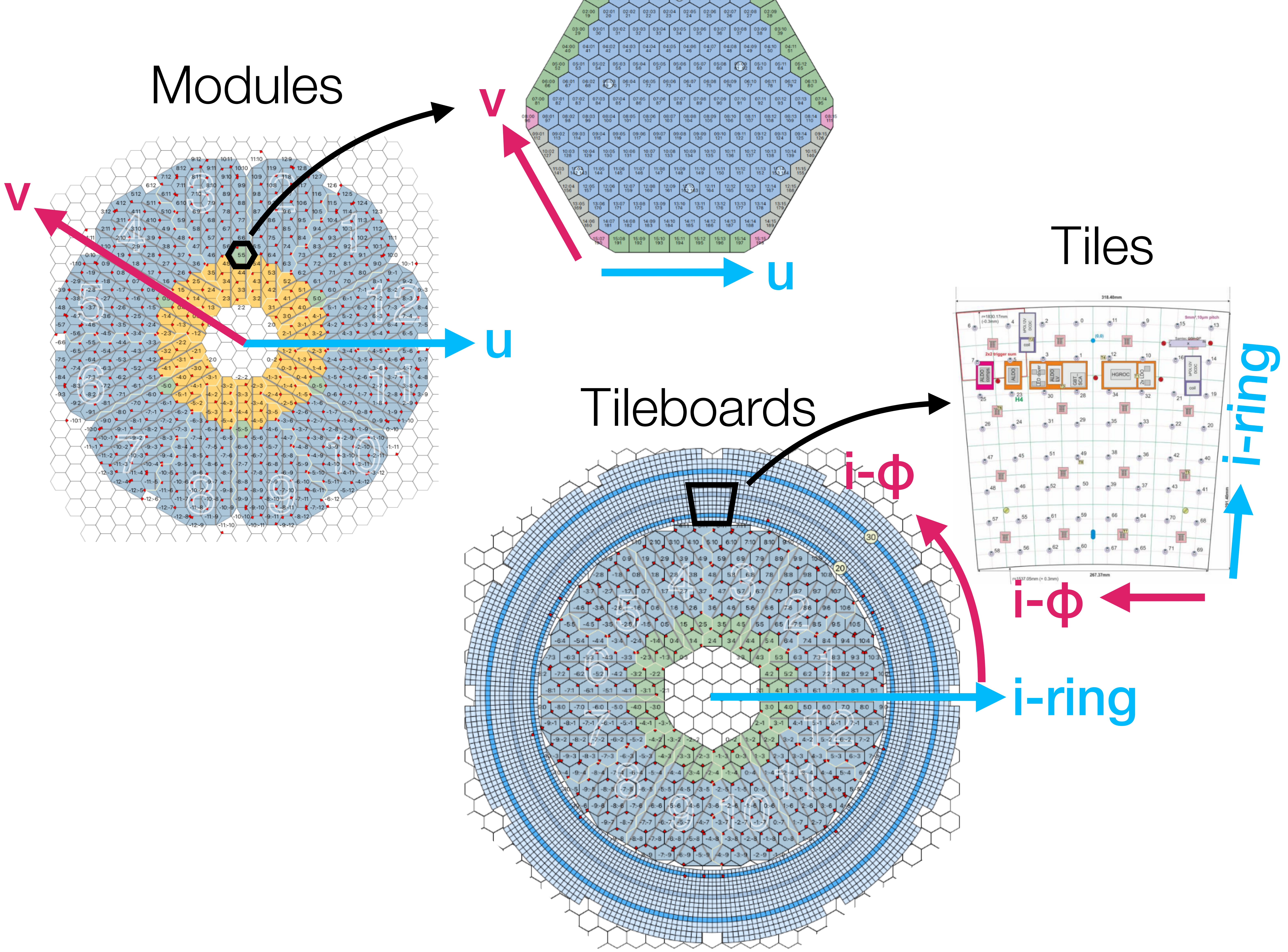
- assign **spatial (geometric) locations** to cells/tiles based on their electronic identifiers
- **essential step** in the RAW data unpacking process



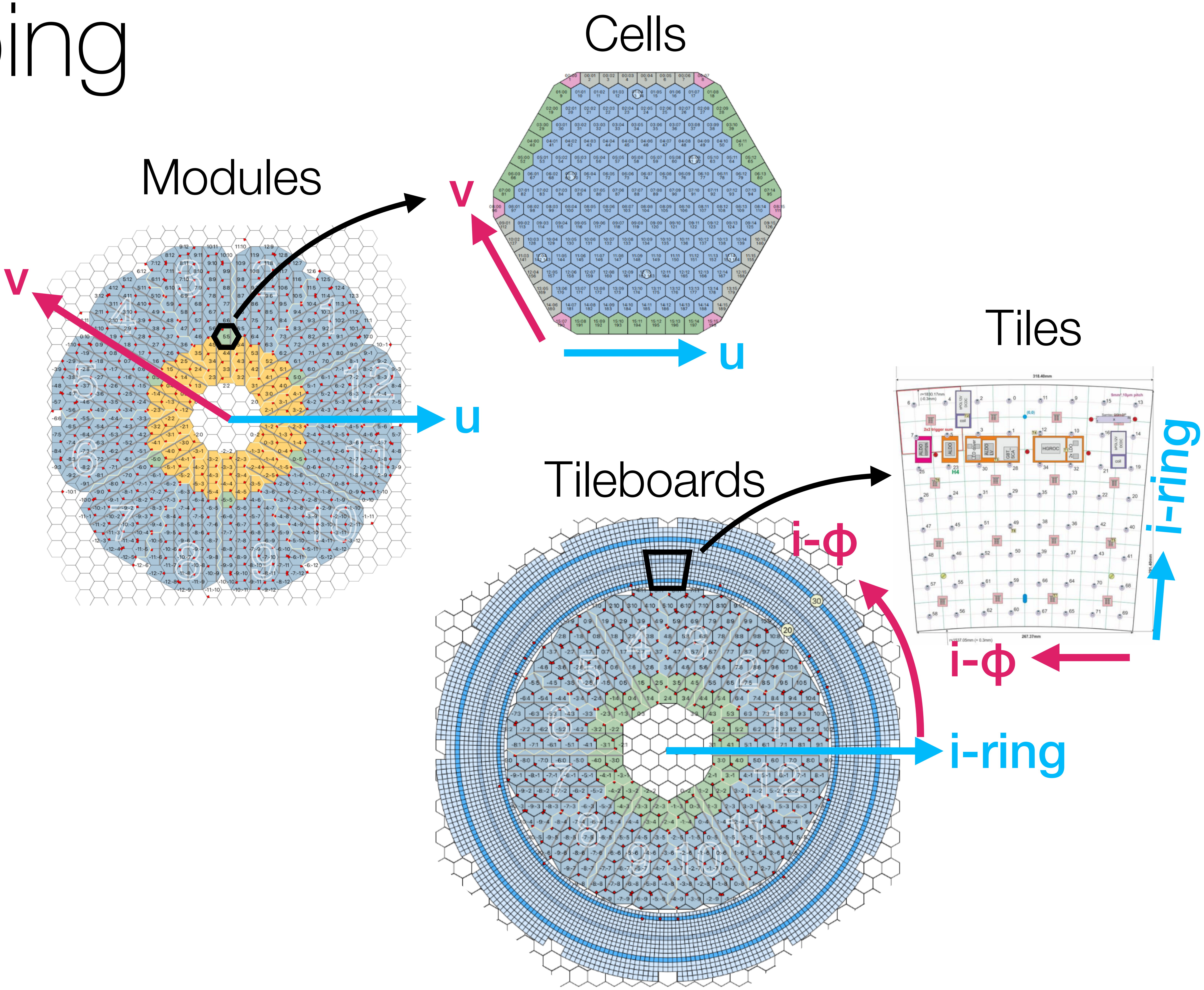
Electronics Mapping

Mapping

- assign **spatial (geometric) locations** to cells/tiles based on their electronic identifiers
- **essential step** in the RAW data unpacking process
- also needed in the **opposite direction** for simulation



Electronics Mapping



Mapping

- assign **spatial (geometric) locations** to cells/ tiles based on their electronic identifiers
- **essential step** in the RAW data unpacking process
- also needed in the **opposite direction** for simulation
- many tricky **“special” cases**:
  - differences between **silicon and scintillator** parts
  - differences **within each part**
  - **partial modules**
  - **calibration/trigger channels**
  - …

Hits Calibration

Calibration

For each hit:

- decide on using **ADC or TOT**
- convert to **raw charge**
  - non-linear effects to be added in the future

raw ADC/TOT

↓

linearization

↓

raw charge q

$$q = \begin{cases} LSB_{ADC} \cdot (ADC + 1/2), & \text{if } T_c T_p = b'00 \\ LSB_{TOT} \cdot (TOT + 1/2) + TOT_0, & \text{if } T_c T_p = b'11 \\ 0, & otherwise \end{cases}$$

Hits Calibration

Calibration

For each hit:

- decide on using **ADC or TOT**
- convert to **raw charge**
  - non-linear effects to be added in the future
- **apply corrections**:
  - channel's pedestal
  - Common Mode (affects the whole module)
  - CM pedestal
  - other correction factors
  - previous bunch crossing corrections (q_{-1})

raw ADC/TOT

↓

linearization

$$q = \begin{cases} LSB_{ADC} \cdot (ADC + 1/2), & \text{if } T_c T_p = b'00 \\ LSB_{TOT} \cdot (TOT + 1/2) + TOT_0, & \text{if } T_c T_p = b'11 \\ 0, & otherwise \end{cases}$$

↓

raw charge q

↓

corrections

$$q_0 = (q - P) + \beta \cdot (CM - P_{CM}) + \kappa \cdot \left[(q_{-1} - P) + (CM - P_{CM})\right]$$

↓

corrected charge q_0

Hits Calibration

Calibration

For each hit:

- decide on using **ADC or TOT**
- convert to **raw charge**
    - non-linear effects to be added in the future
- **apply corrections**:
    - channel's pedestal
    - Common Mode (affects the whole module)
    - CM pedestal
    - other correction factors
    - previous bunch crossing corrections (q_{-1})
- convert **charge to energy**:
    - scale q_0 to N_{MIP}
    - correct for irradiation (per channel)
    - scale to units of energy

raw ADC/TOT

↓

linearization

$$q = \begin{cases} LSB_{ADC} \cdot (ADC + 1/2), & \text{if } T_c T_p = b'00 \\ LSB_{TOT} \cdot (TOT + 1/2) + TOT_0, & \text{if } T_c T_p = b'11 \\ 0, & otherwise \end{cases}$$

↓

raw charge q

↓

corrections

$$q_0 = (q - P) + \beta \cdot (CM - P_{CM}) + \kappa \cdot \left[(q_{-1} - P) + (CM - P_{CM})\right]$$

↓

corrected charge q_0

↓

conversion (q0 → N_{MIP} → E)

$$MIP_{scale} = CCE \cdot MIP_{scale,uniraddiated}$$

$$N_{MIP} = \frac{q_0}{MIP_{scale}}$$

$$E = \langle dE/dx \rangle \cdot \alpha \cdot N_{MIP}$$

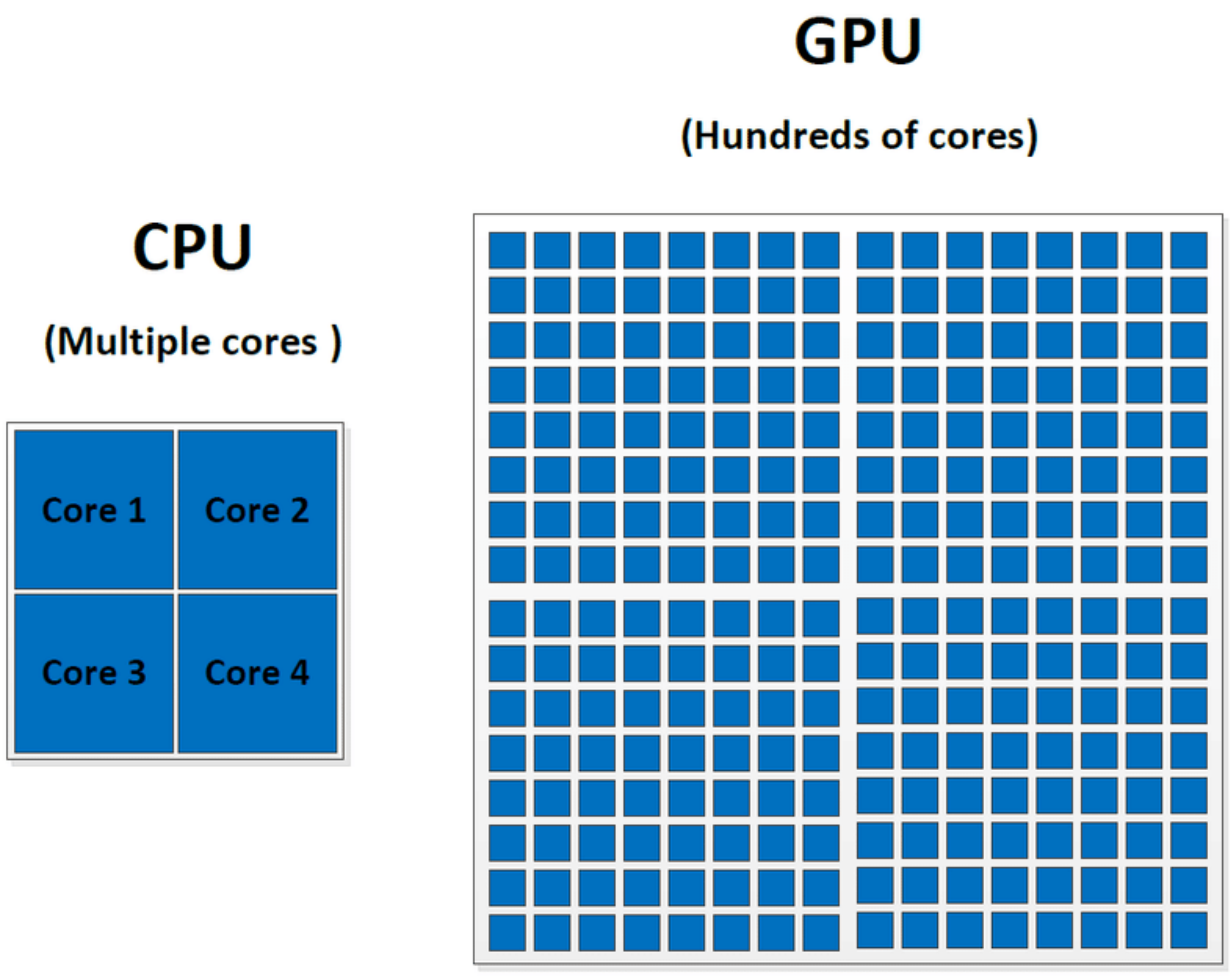
↓

energy E

Heterogenous Computing

Heterogeneous Computing

- using **more than one type** of processing unit, e.g. CPU+GPU
- especially interesting for us — **GPUs**:
  - very large **parallelization** → perfect for the **same operations** applied to **millions of hits**



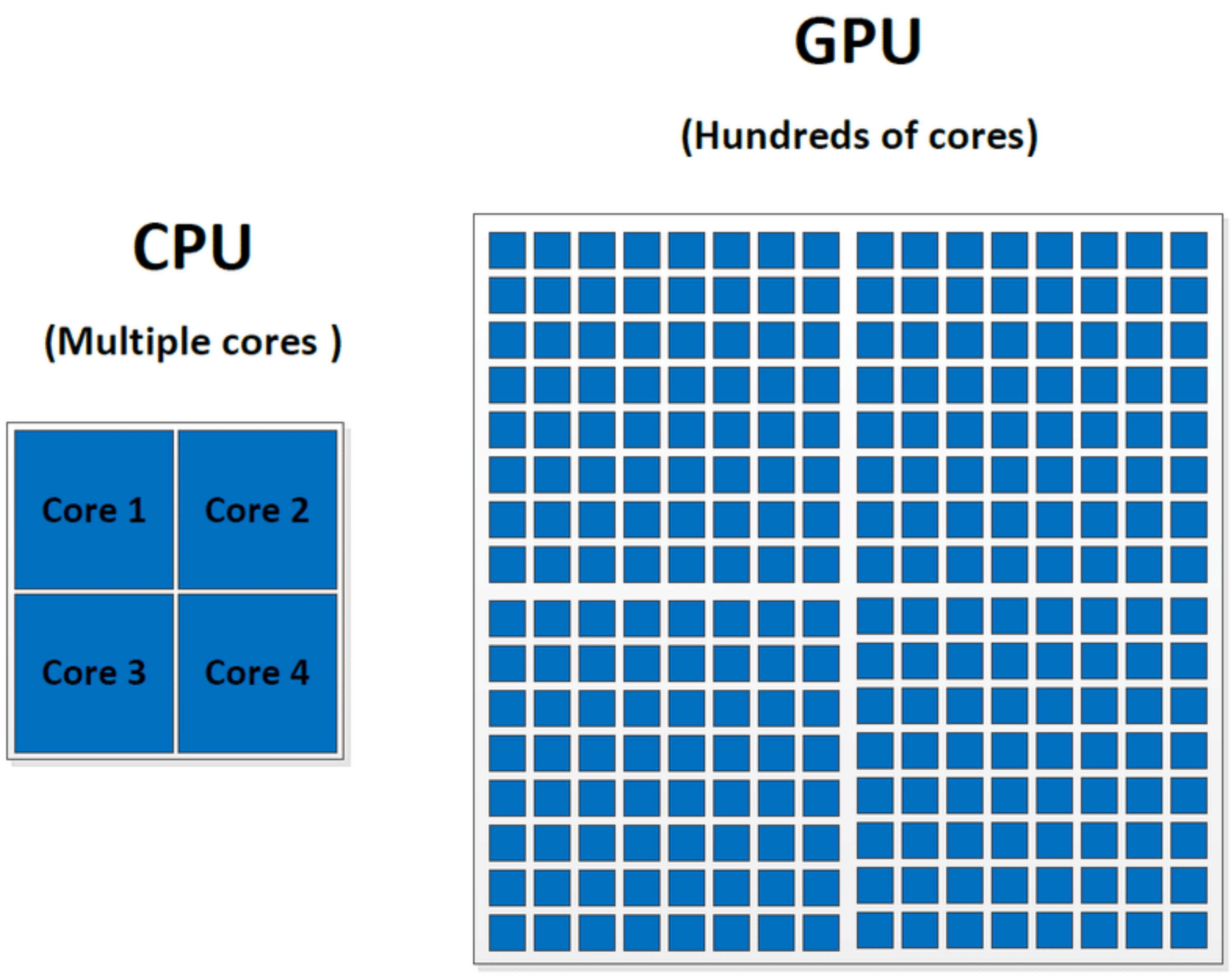
Heterogenous Computing

Heterogeneous Computing

- using **more than one type** of processing unit, e.g. CPU+GPU
- especially interesting for us — **GPUs**:
  - very large **parallelization** → perfect for the **same operations** applied to **millions of hits**

Alpaka — Abstraction Library for Parallel Kernel Acceleration

- **portable, performant code** across various types of accelerators:
  - multi-core CPUs
  - GPUs
  - std::thread
  - and others…
- **back-ends** for:
  - CUDA
  - OpenMP
  - Boost.Fiber
  - and many others
- **transparent** for the coder (all programmed in the same way)



Results

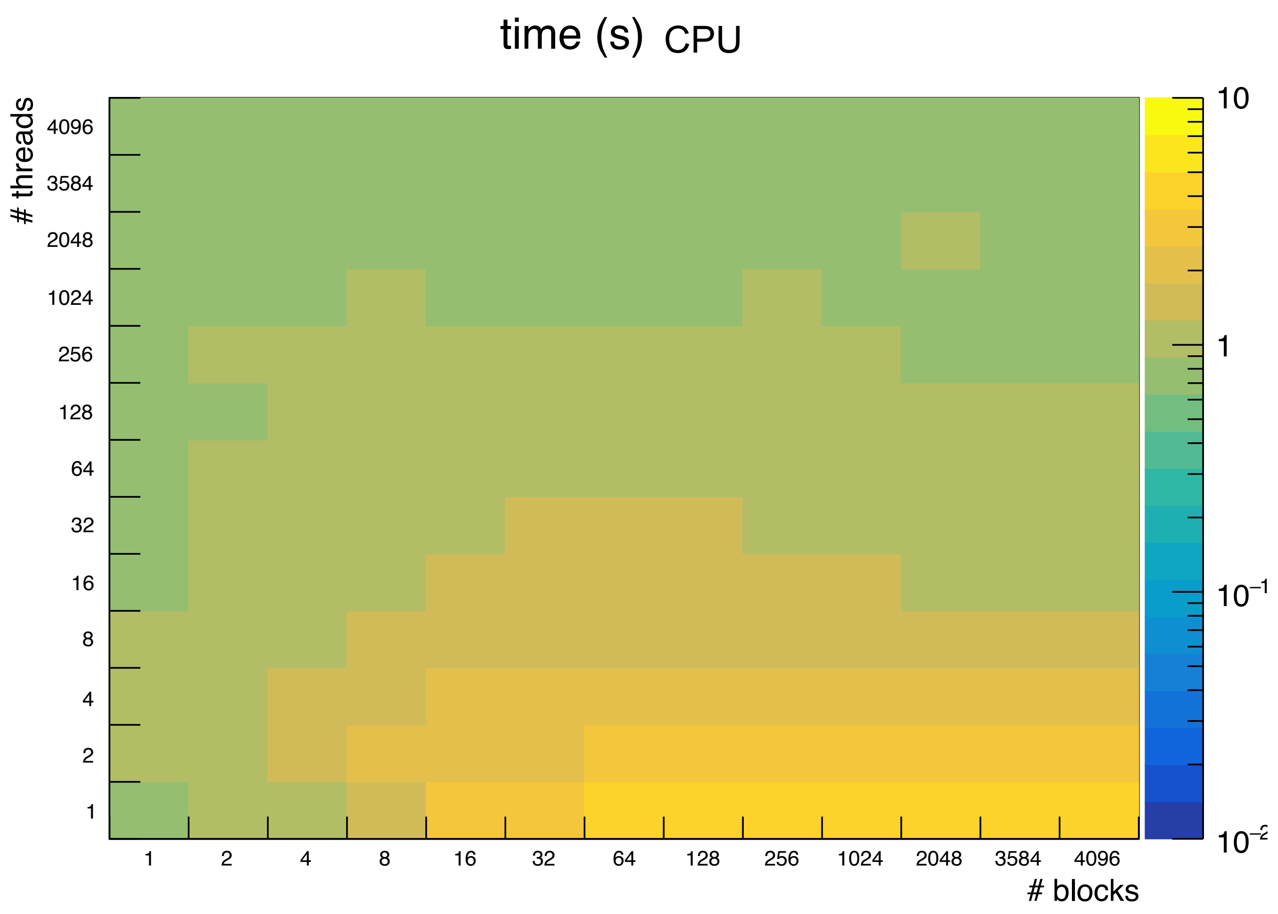
Calibration speedup

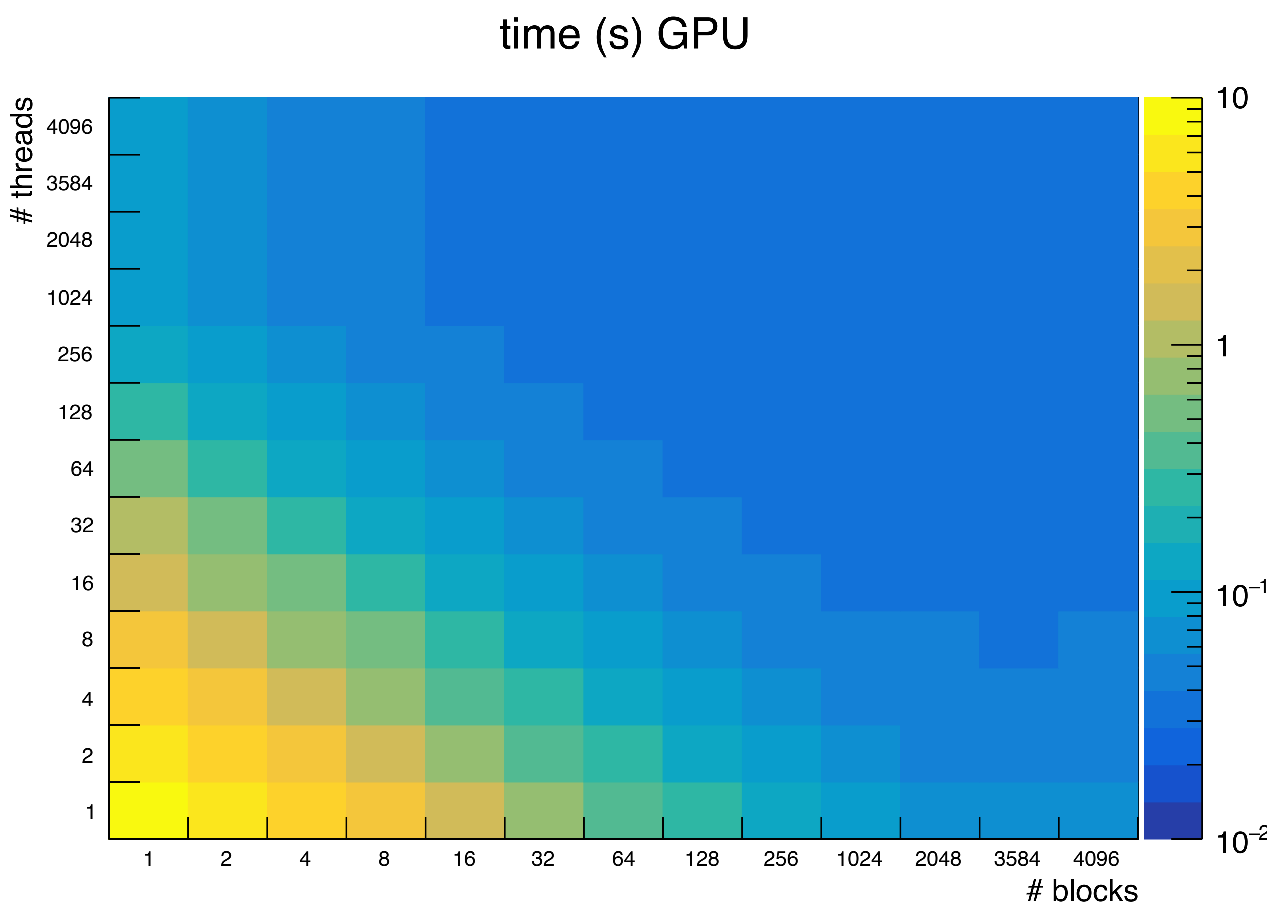
- to asses performance
  → take simulated events and **clone the hits** in them many many times

Results

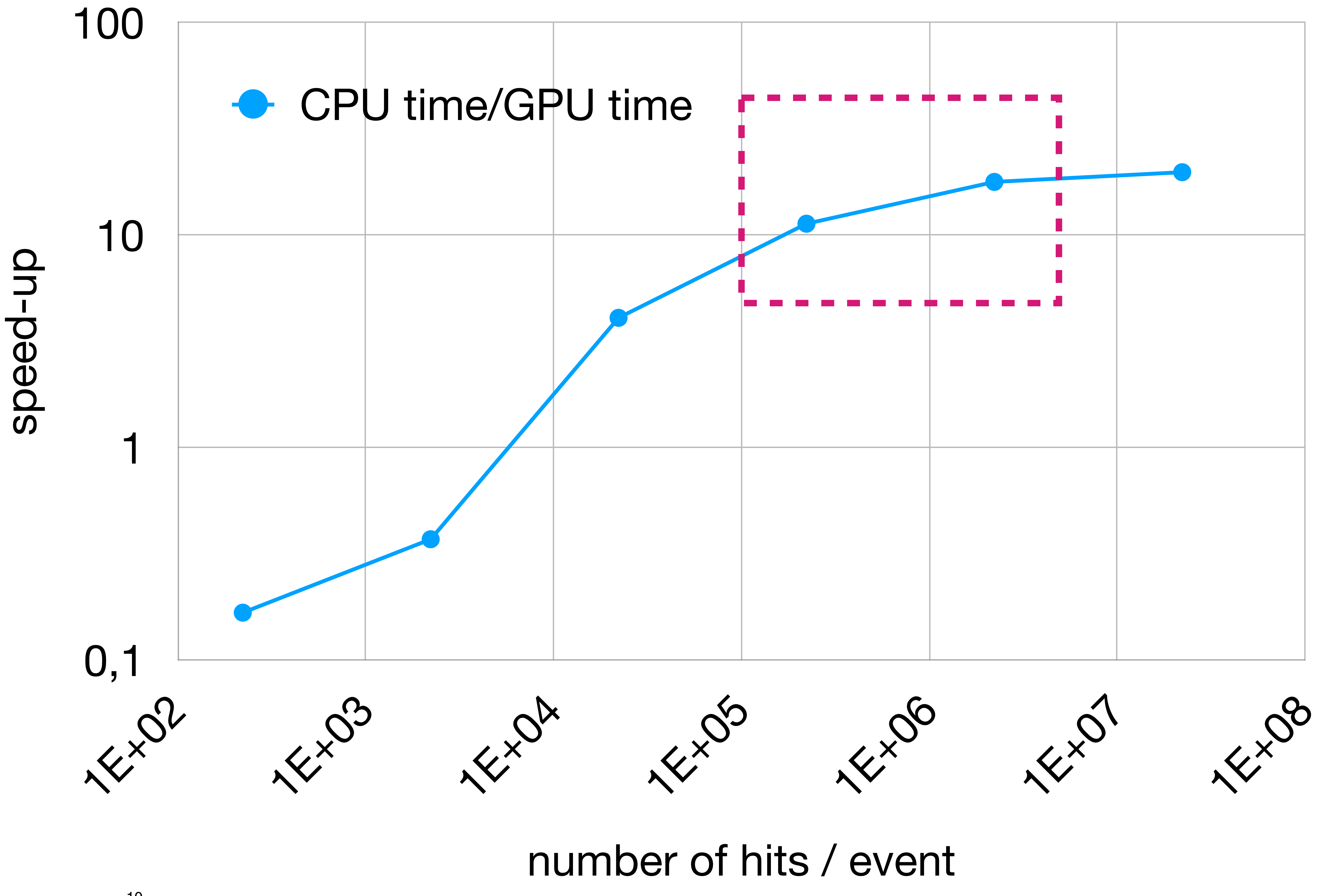
Calibration speedup

- to asses performance
  → take simulated events and **clone the hits** in them many many times
- number of **threads/blocks optimized** separately for CPU and GPU



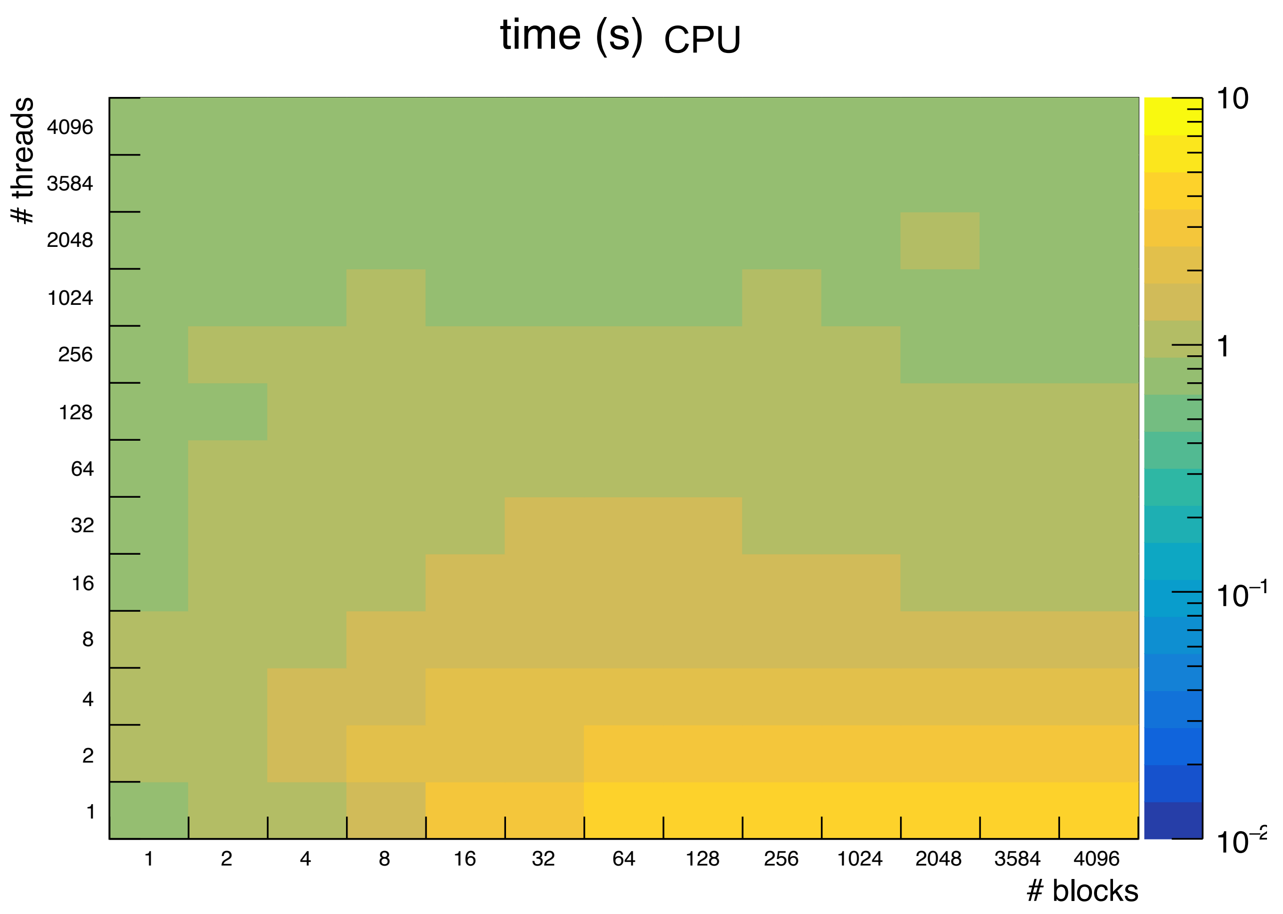


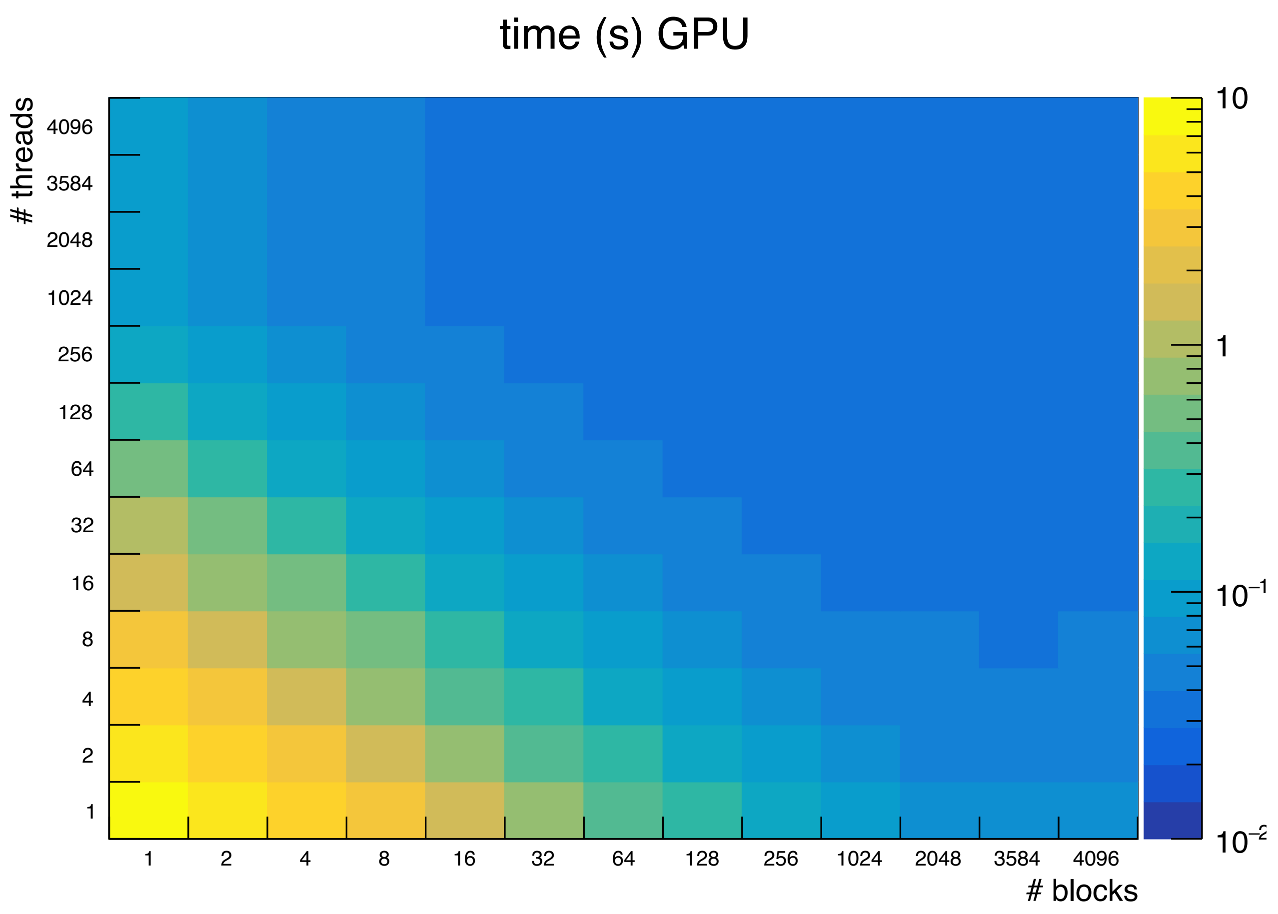
Results



Calibration speedup

- to asses performance
  → take simulated events and **clone the hits** in them many many times
- number of **threads/blocks optimized** separately for CPU and GPU
- in the region expected for real busy events: **10x speedup!**





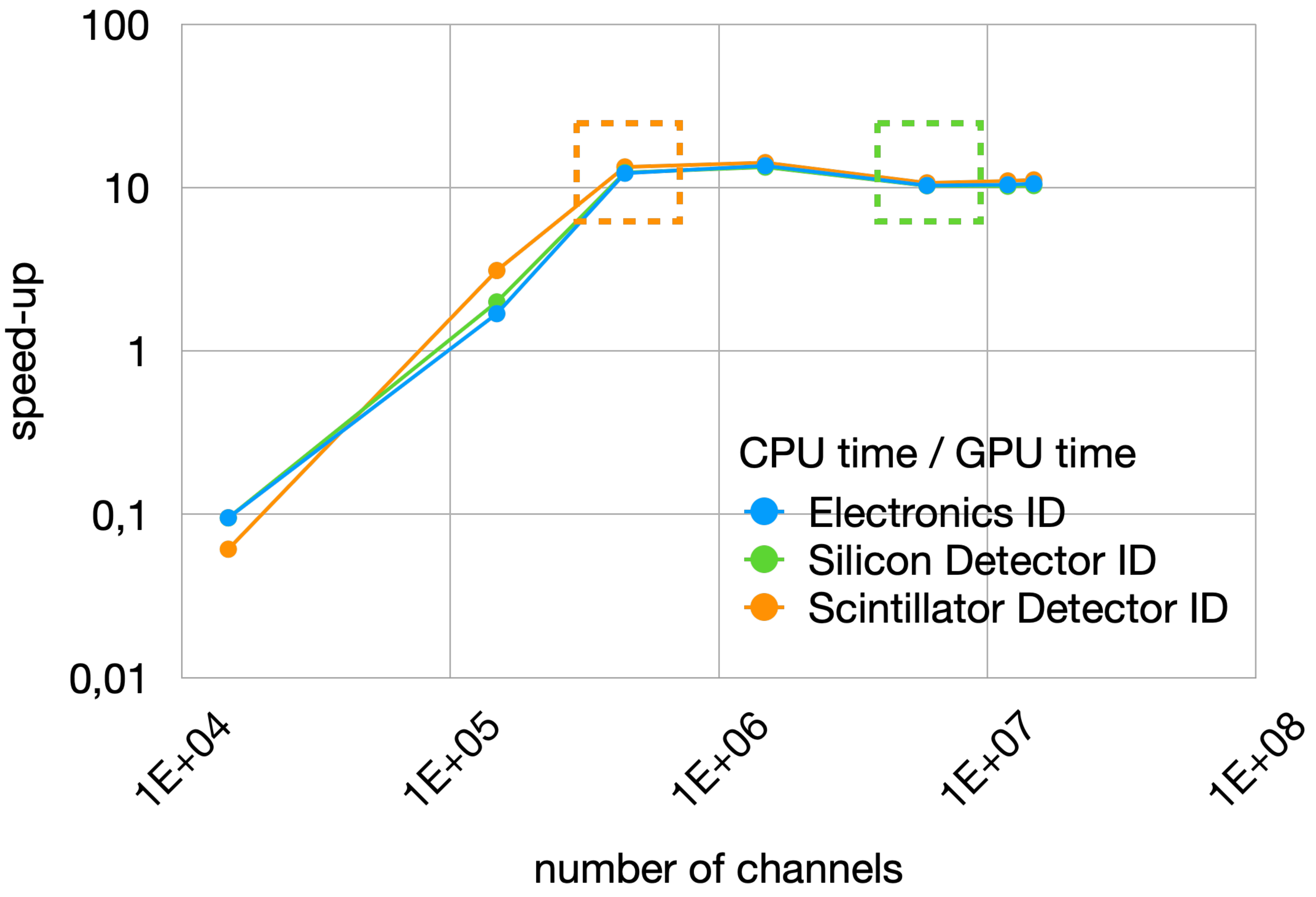
Results

Mapping speedup

- mapping implemented using **alpaka data structures** storing:
  - **electronics information** in Electronics IDs
  - **geometrical information** in Silicon or Scintillator Detector IDs

Raw value	Method	Number of channels	Time to retrieve
Electronics ID	module eleid + cell eleid	6 000 000	0.7 μs
Silicon Detector ID	module detid + cell detid	6 000 000	0.3 μs
Scintillator Detector ID	getSiPMDetectorID(zside, plane, module u, celltype, cell u, cell v)	390 000	2ms

Results



Mapping speedup

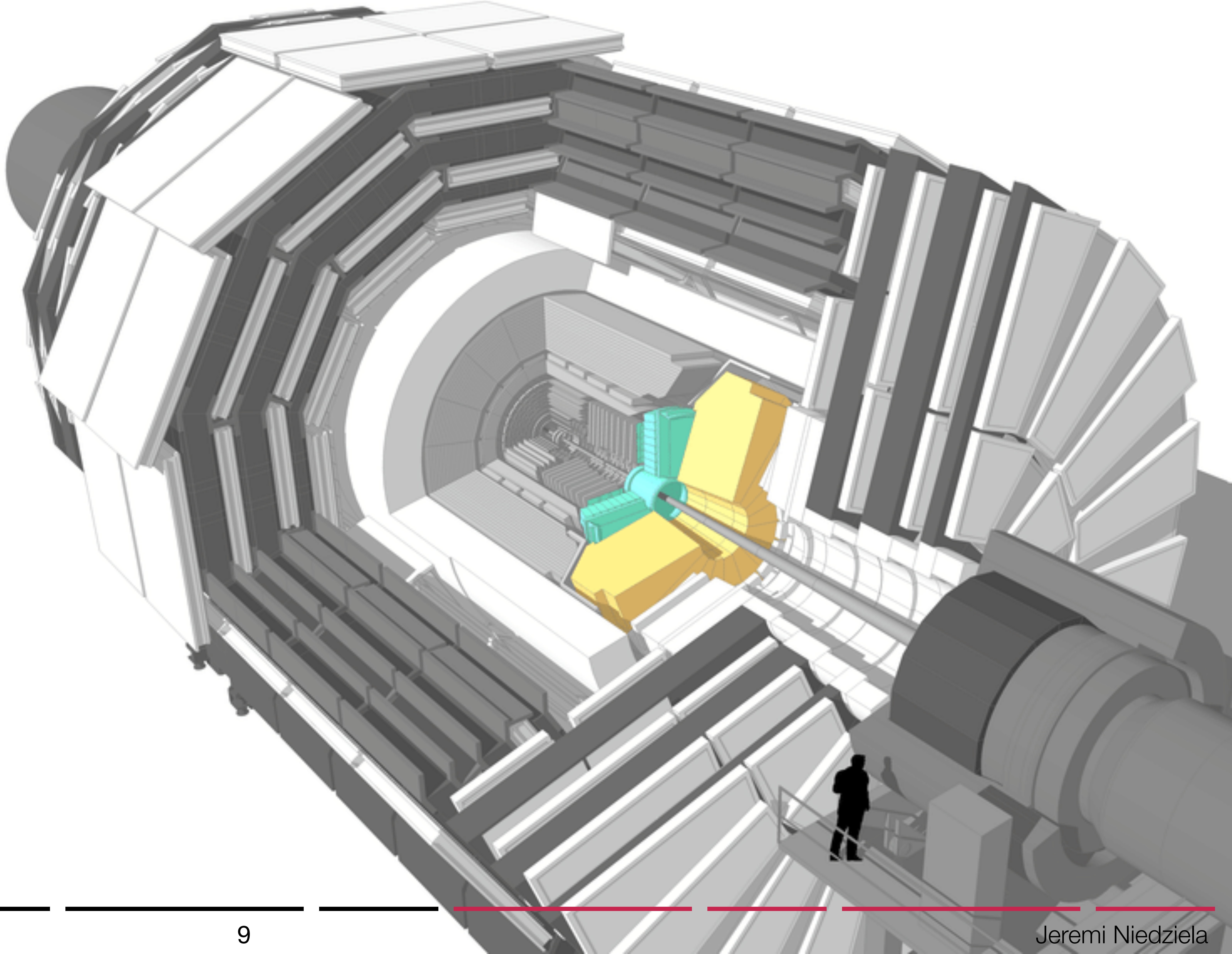
- mapping implemented using **alpaka data structures** storing:
  - **electronics information** in Electronics IDs
  - **geometrical information** in Silicon or Scintillator Detector IDs
- calculations of IDs during **setup can be run on GPUs**
- For **390k Scintillator channels**: **10x speedup**
- For **6M Silicon channels**: **10x speedup**

Raw value	Method	Number of channels	Time to retrieve
Electronics ID	module eleid + cell eleid	6 000 000	0.7 μs
Silicon Detector ID	module detid + cell detid	6 000 000	0.3 μs
Scintillator Detector ID	getSiPMDetectorID(zside, plane, module u, celltype, cell u, cell v)	390 000	2ms

Summary

Key points

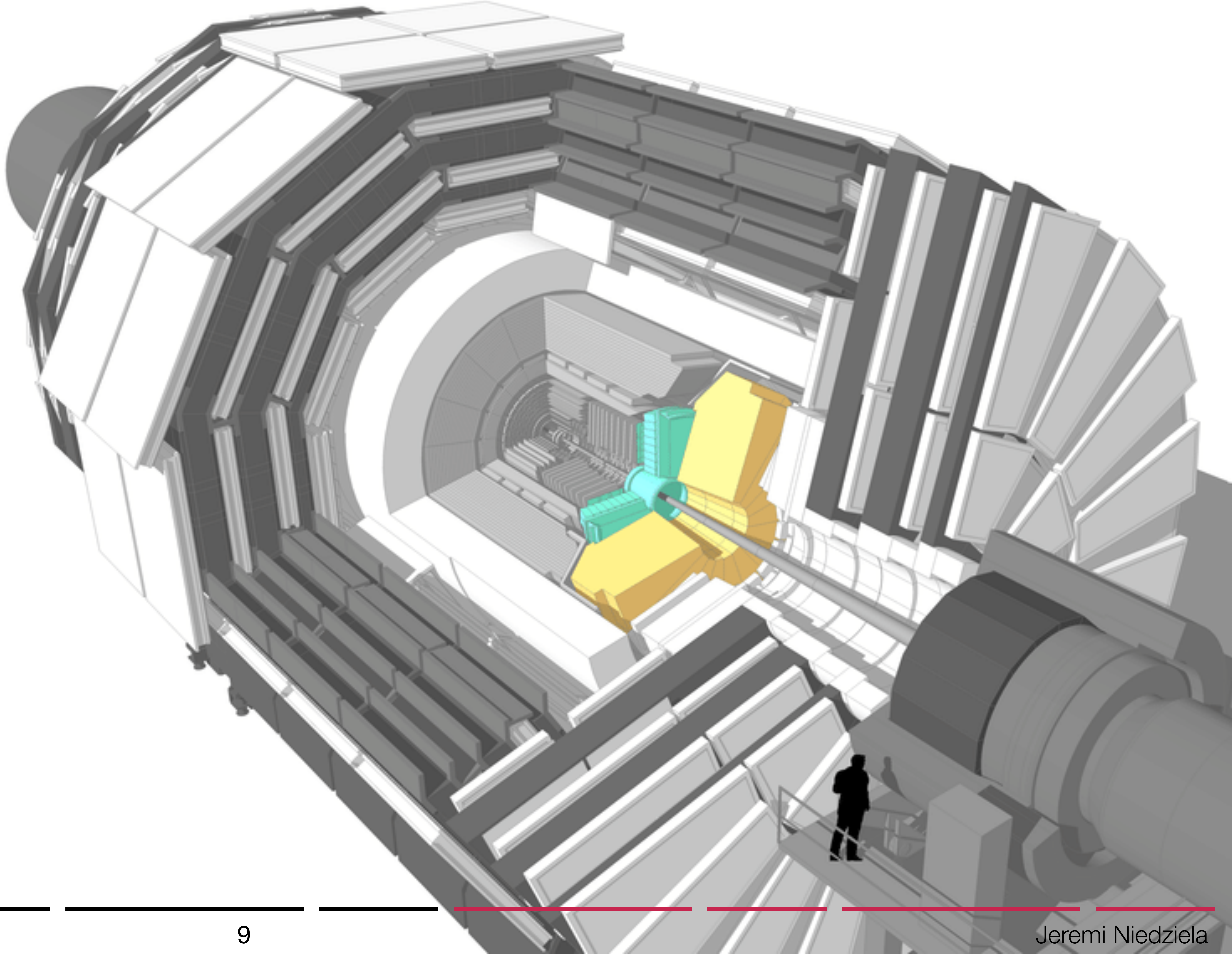
- HGCal calibration and reconstruction are challenging
- **Heterogenous Computing** helps to address these challenges
- Alpaka already provides **10x speedup of calibration**
- **10x speedup in mapping** setup as a byproduct of making it alpaka-ready



Summary

Key points

- HGCal calibration and reconstruction are challenging
- **Heterogenous Computing** helps to address these challenges
- Alpaka already provides **10x speedup of calibration**
- **10x speedup in mapping** setup as a byproduct of making it alpaka-ready



Future Prospects

- started work on **DQM**
- test calibration in a **more realistic setting** (test beam data)
- make sure we meet **timing requirements**
- adjust calibration for the SiPM part of HGCal
- introduce **Alpaka in other parts** of the reconstruction chain