

Computing in HEP

Concezio Bozzi

INFN Sezione di Ferrara

Monopoli, October 3rd 2025



XXXV International School
“Francesco Romano”

on Nuclear, Subnuclear and Astroparticle Physics

DISCLAIMER

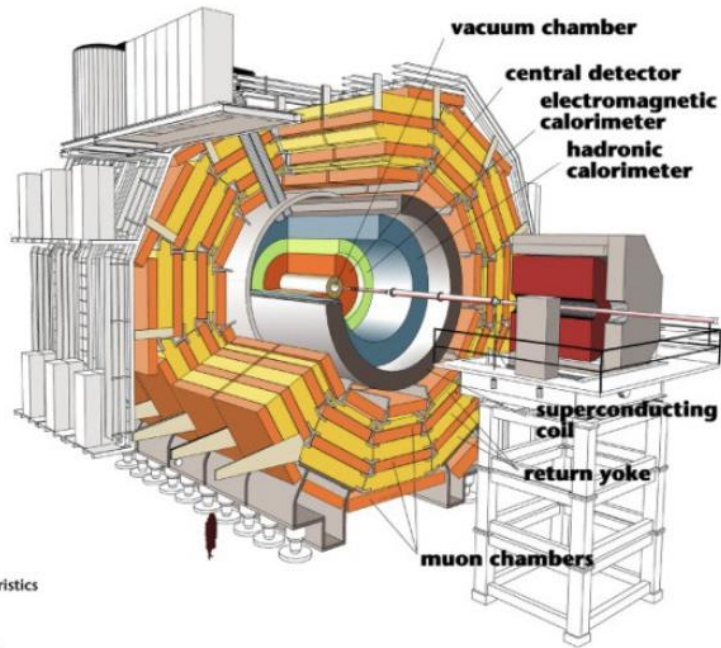
Computing in HEP *at the LHC*

Concezio Bozzi

INFN Sezione di Ferrara

Monopoli, October 3rd 2025

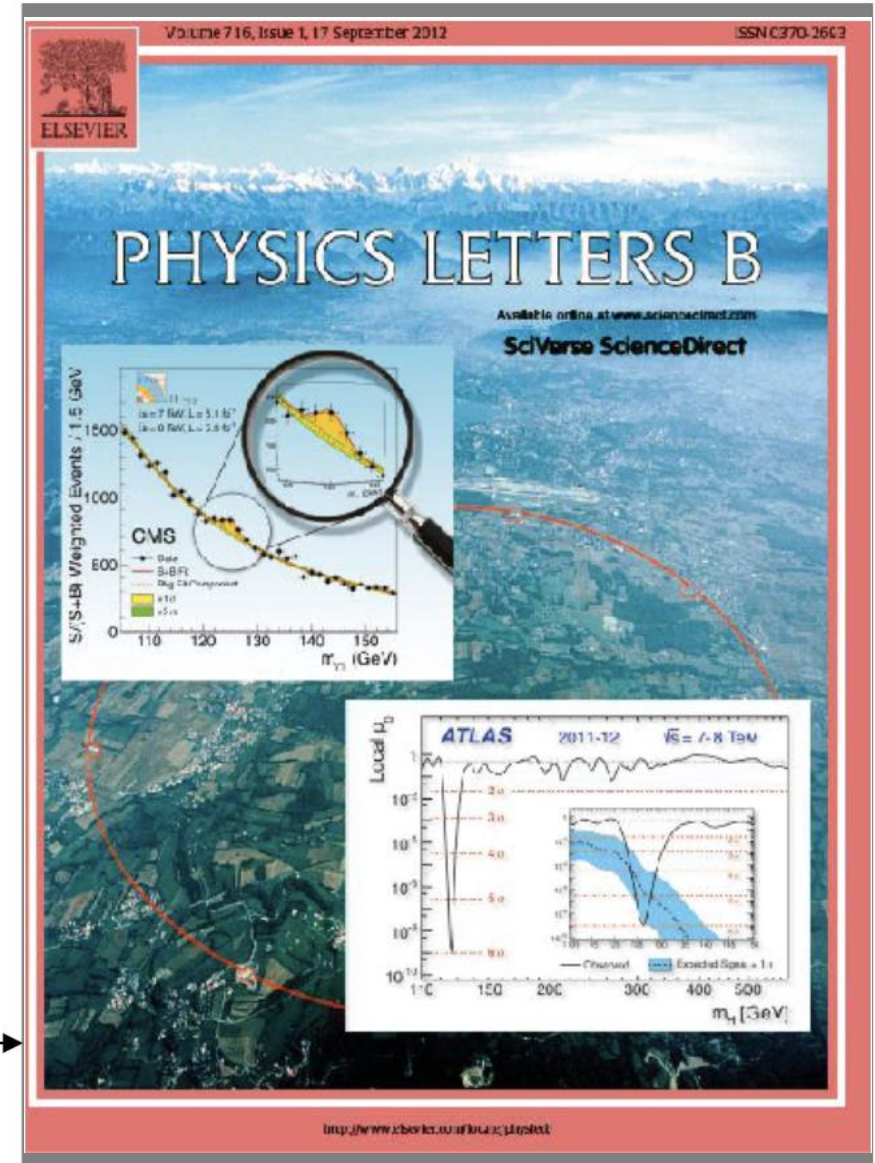
...Many of the concepts and topics that will be discussed (hardware & software infrastructures, technologies) are relevant also for Other HEP (Belle-2, ...), neutrino (DvNE, ...), astroparticle (SKA, CTA, GW, ...) experiments



Detector characteristics

Width: 22m
Diameter: 15m
Weight: 14'500t

```
Paper paper15
Data higgsdata
...
paper15=make_paper(higgsdata)
...
```

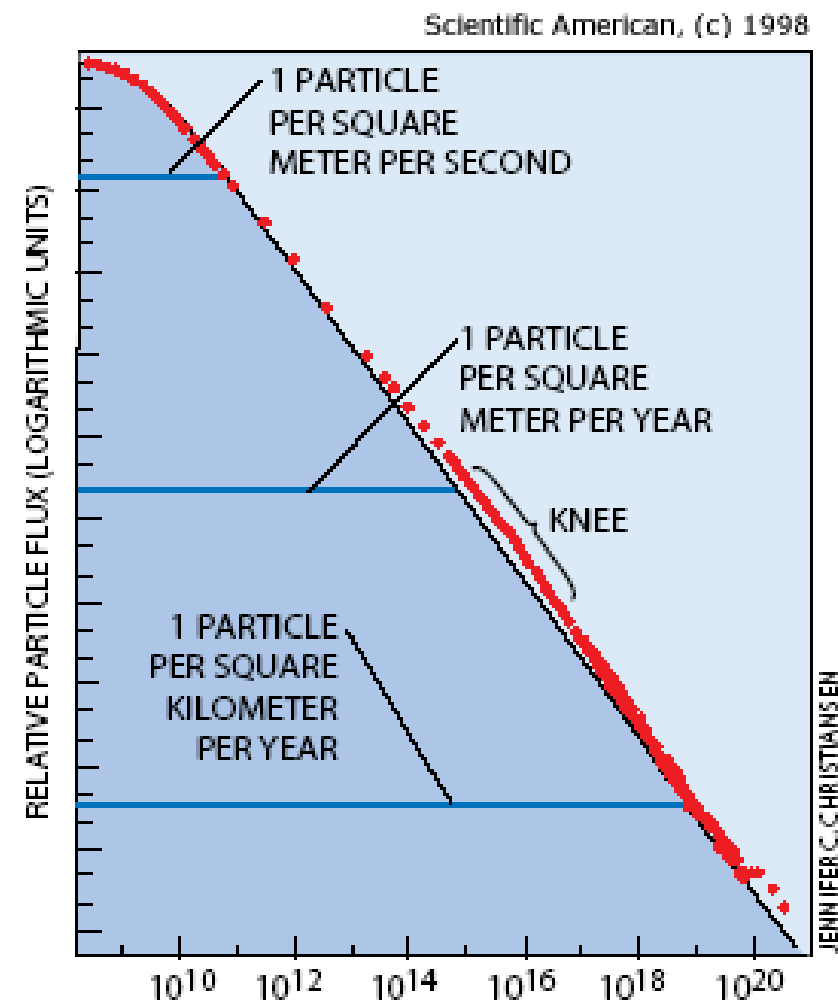


Outline

- The relevance of computing in HEP
- From RAW data to physics results
- Software and computing infrastructure
- Future evolution

Why computing is relevant in HEP

- Current HEP research needs to look into **high energy** and/or **rare** processes and/or **very precise** measurements
 - High Energy: Look up in the sky!
 - Astroparticle Physics, the universe produces for you cosmic rays (measured up) to some 10^{21} eV (10^9 TeV)
 - But they are rare!
 - Rare & Precision: Produce (a lot of) high energy events using colliders
 - Current best is “only” at 14 TeV (c.m.)
 - but we can produce billions per second
- The need for a lot of computing is an unfortunate consequence



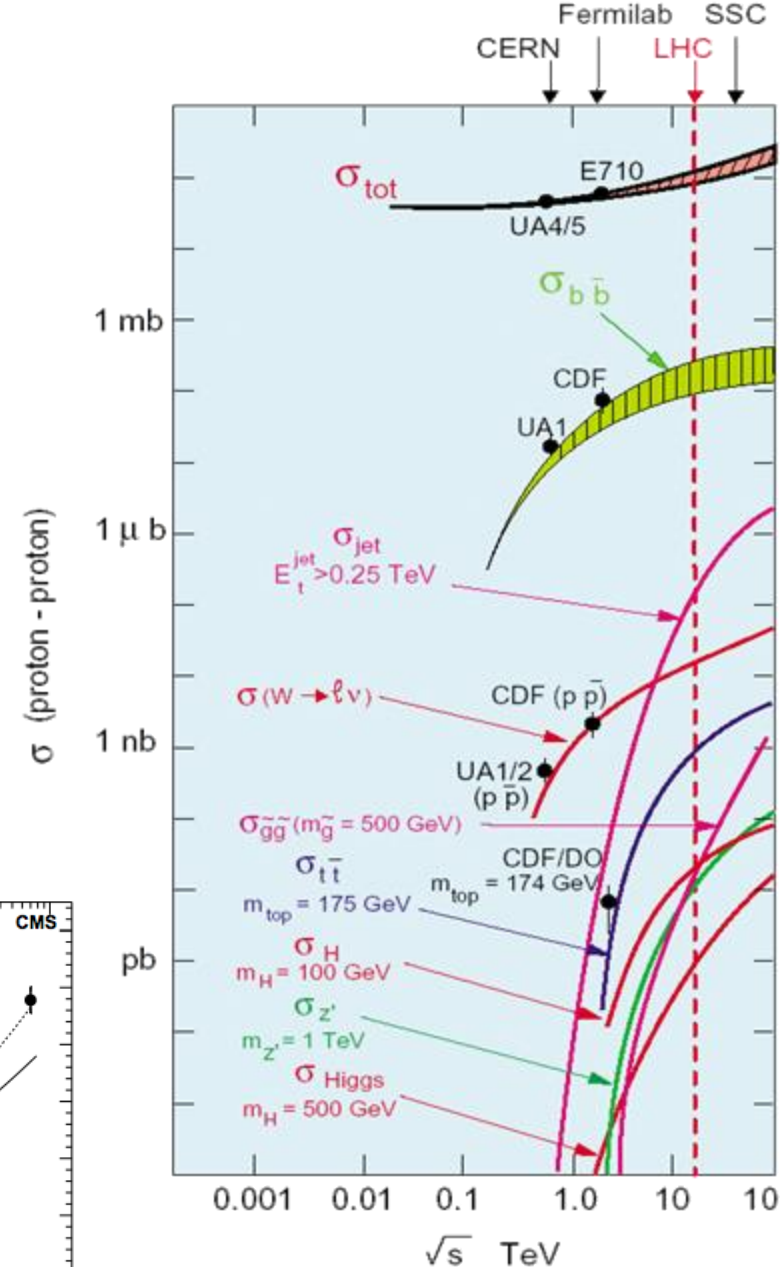
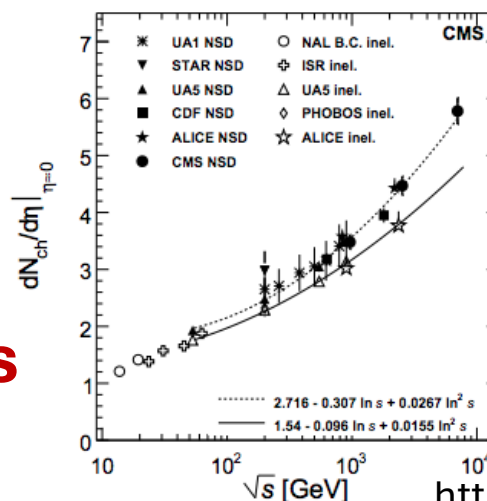
Why computing is relevant in HEP

- Most of the reasoning involves the relation between the cross section of a given process and the number of events generated

$$N = \sigma \times L_{\text{int}}$$

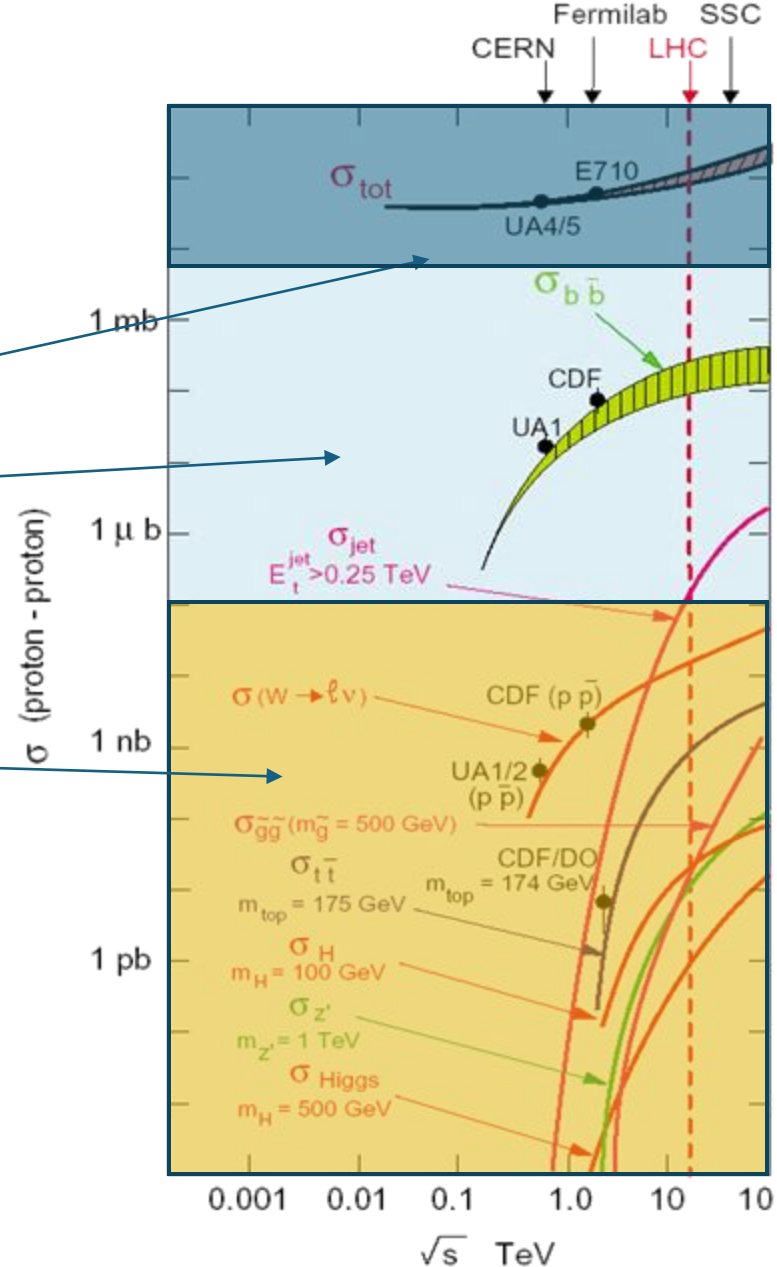
- More (c.m.) energy in the collision of beams: the total cross section increases + the complexity of the collision results increases

- more, and more crowded events**



Why computing is relevant in HEP

- This part of the cross section plot is “mostly understood and not interesting”
- This part is “interesting”, as it has large cross-section and is sensitive to new physics through precision measurements of (rare) heavy quark flavour processes
- This part is “interesting”, but has cross sections up to billion times smaller
- Unfortunately quantum mechanics tells us the “choice of the process” is completely probabilistic: you cannot force nature to produce only what you care for
- **In order to produce the latter two, you need to produce (a lot of) the former**



Total number of “trials” needed

- Take ATLAS / CMS as an example
- For a cross section of 10^5 fb, to produce 10.000.000 Higgs in 5 years (per experiment) one needs

$$L_{\text{int}} = 100 \text{ fb}^{-1} \text{ integrated luminosity } (10^7 / (10^5 \text{ fb}))$$

- This translates into an instantaneous lumi(*)

$$L_{\text{INST}} = 10^{41} \text{ cm}^{-2} / (5 \text{ y} * 3 * 10^7 \text{ s/y} / 5_{(\text{ineff})}) = O(10^{34}) \text{ cm}^{-2} \text{ s}^{-1}$$

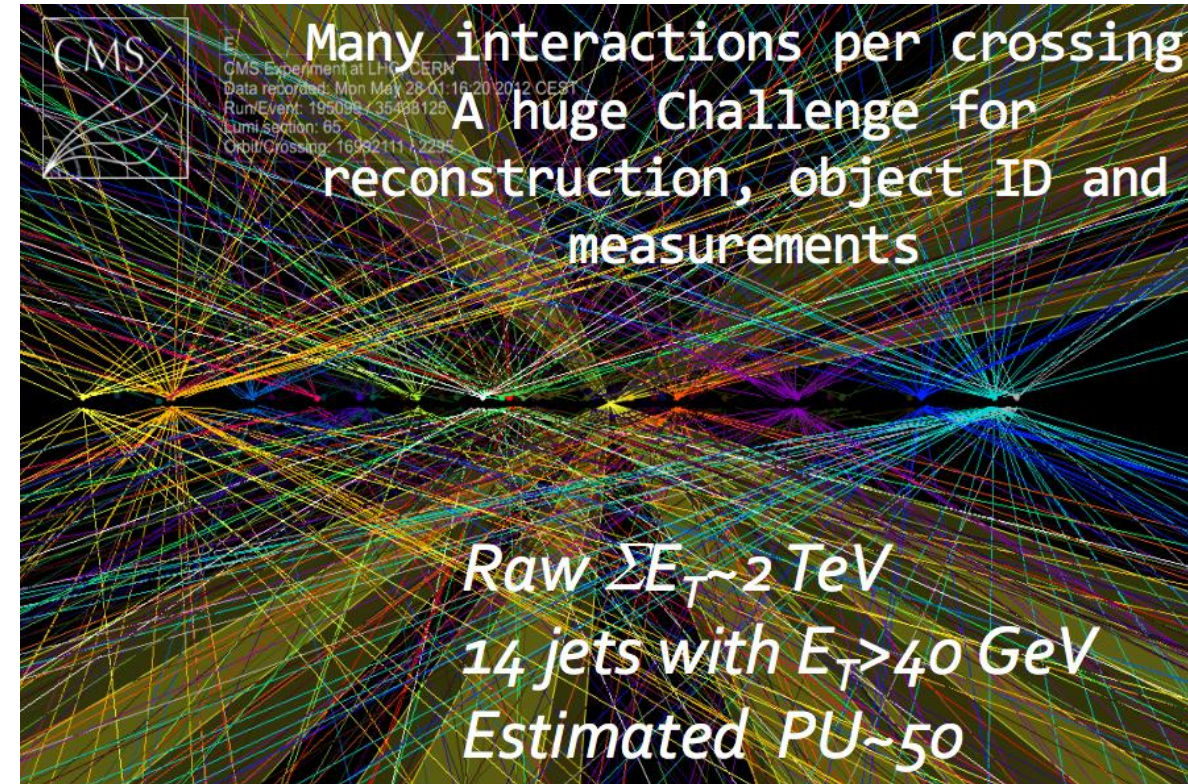
The LHC!

- .. But at the same time, 100 fb^{-1} will result in some 10^{16} «uninteresting» collisions

(*) assuming an efficiency factor ~ 5 for shutdown periods, vacations, repairs, etc, and noting that $1 \text{ b} = 10^{-24} \text{ cm}^2 \rightarrow L_{\text{int}} = 100 \text{ fb}^{-1} = 10^{41} \text{ cm}^{-2}$

Selecting the interesting collisions

- Not an easy task, they do not always look so different
- On top of this, the 25 ns bunched structure of LHC superimposes several proton collisions in a single bunch crossing (~30-50 Run-2, up to 80 Run-3, up to 200 in the future), and most of the signals come from the uninteresting one (and, they are not colored in the figure on the right!)
 - An online selection is not trivial; in order to have decent efficiency on the “interesting events” you cannot be too picky
 - In the flavour sector, even the interesting events are a lot



Back-of-the-envelope estimate of storage needs

- Simplified model for “a detector”
 - take a “**picture**” of a collision every 25 ns (**40 MHz**)
 - **O(100) Million** detector channels (“pixels”)
 - Assume 1 channel = 1 byte
 - the data rate would be
$$40\text{e6 ev/s} * 100\text{e6 byte/ev} = 4 \text{ PB/s}$$
- A “**storage problem**” is automatic given the needs for looking into rare events with an high precision



A data deluge!

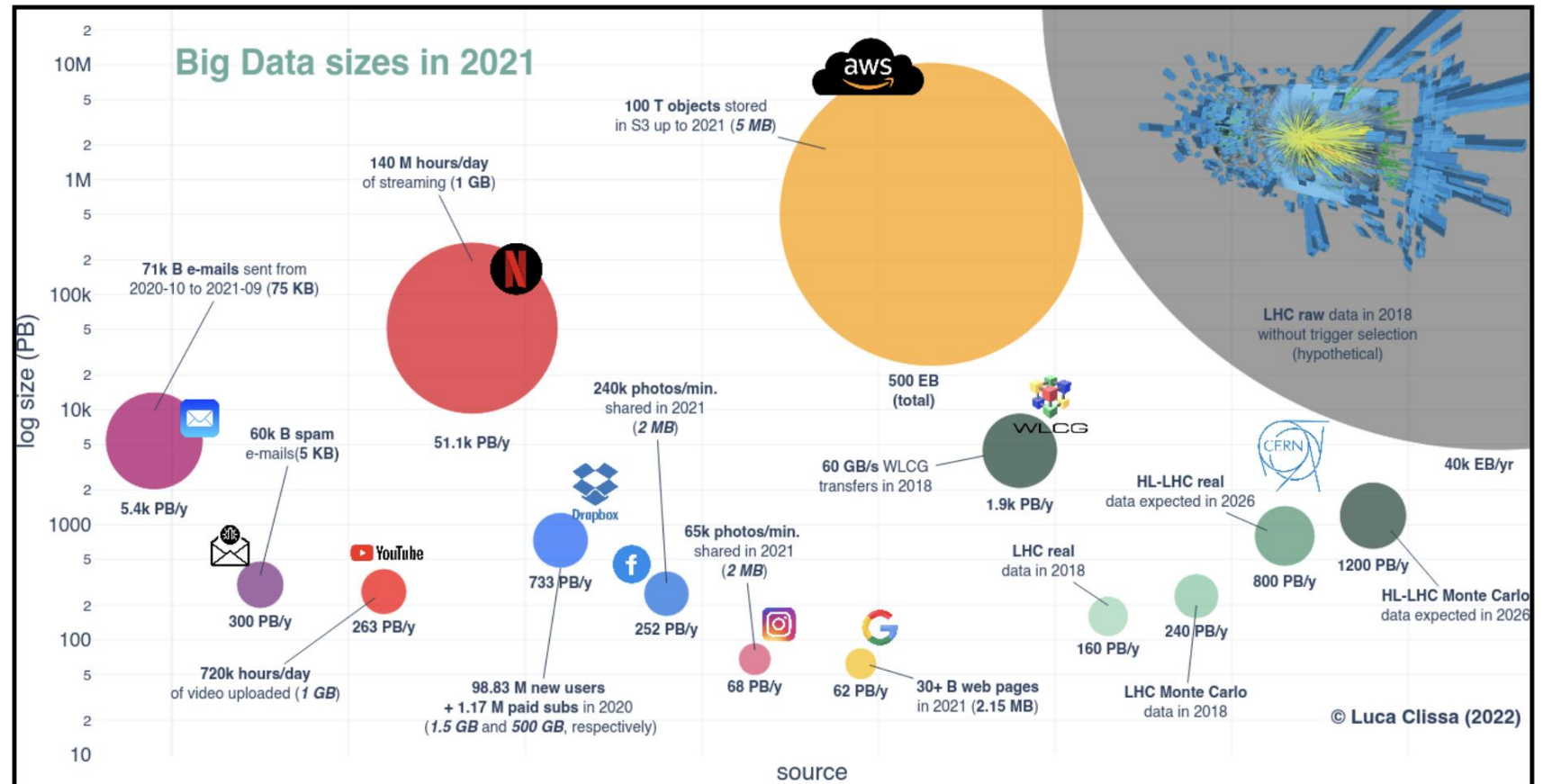
- In an ideal world, all the 40 MHz 25 ns snapshots would be saved and analyzed
- ...but 4 PB/s in 5 years would be 120 ZB (ZettaBytes = 10^{21} bytes)
- We cannot save 4 PB/s for any reasonable number of seconds, and the experiments need to last for years; hence a number of solutions / tricks / approximations needs to be found
 - Easy ones: Zero suppression: do not save the reading of channels which are not “significant” (lossy compression): 100 MB/ev \rightarrow 1 MB/ev
 - Complex ones: try and interpret the events as they flow, and select “enough of the interesting ones” \rightarrow the trigger
- In practice, a much lower rate is saved for \$\$ reasons
 - years of studies have defined the “minimum” possible while still preserving the physics capabilities at least for the most important physics channels.

In context

HEP produces huge datasets

- Comparable to industry applications
- with different usage patterns
- By using public funding

RAW data rates are totally unprecedented



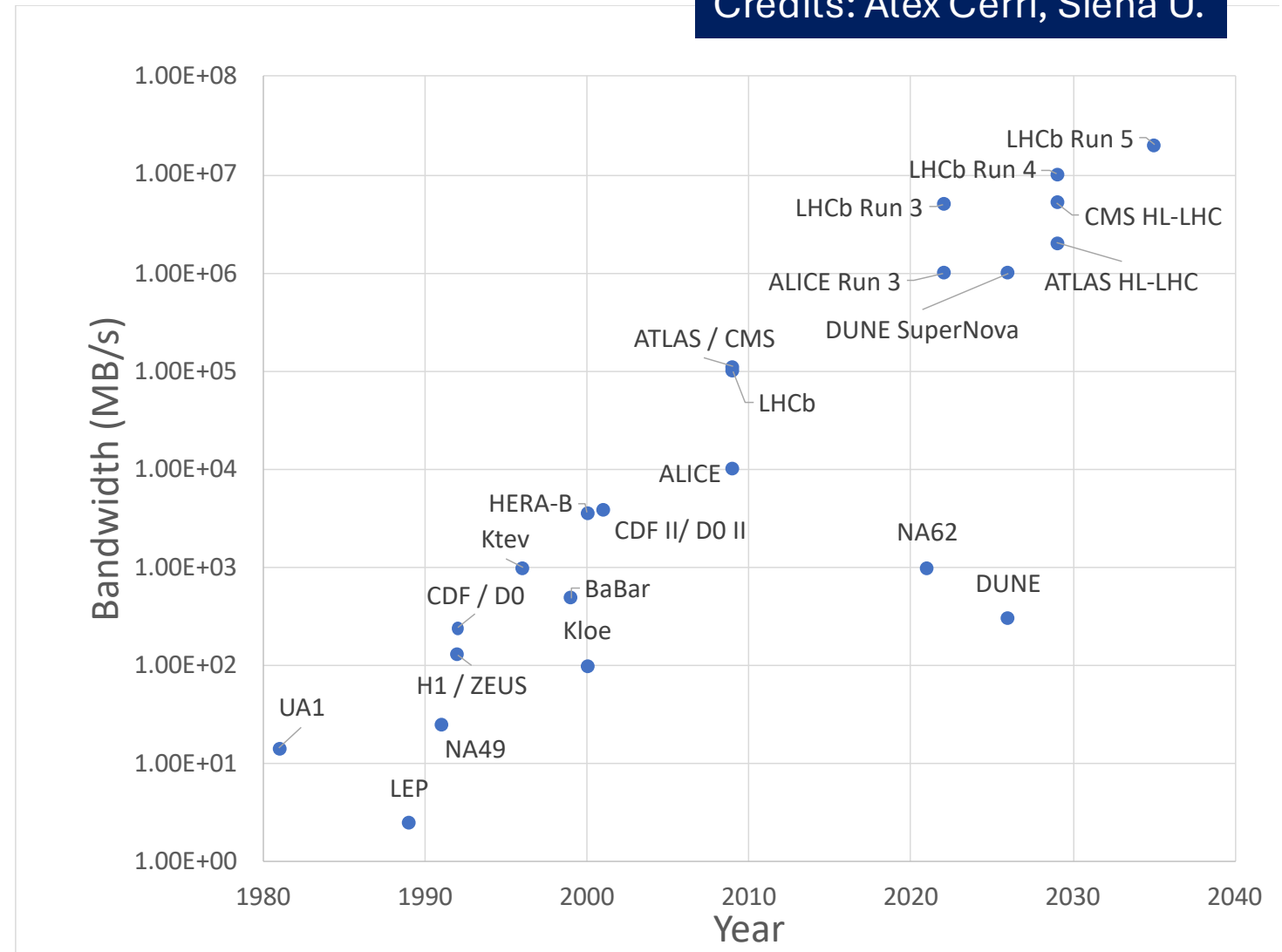
Challenging

Data throughput **from detector back-ends** today: 1-10TB/s

Typical LHC “live-time”: 7Ms/year

→ Data volumes: 7-70EB/year

Credits: Alex Cerri, Siena U.



The limiting factor of a HEP experiment

- Apart from some limits on the electronics (“*I cannot dispatch more than X consecutive triggers*”), **the real limit** on the numbers and type of events collected by HEP experiments **is the Computing**, and on its turn the **amount of money** one can dedicate to that.
- If you want, it is a reversed process: *I know what I can spend on the computing* → *I know how many events I can collect* → *I know what type of physics I can do.*
- **This is why any R&D, new idea, new solution which allows to reduce the Computing costs, is very visible and increases the physics potential of the experiments**

(S. Roiser)

CPU, Disk, Tape And All That



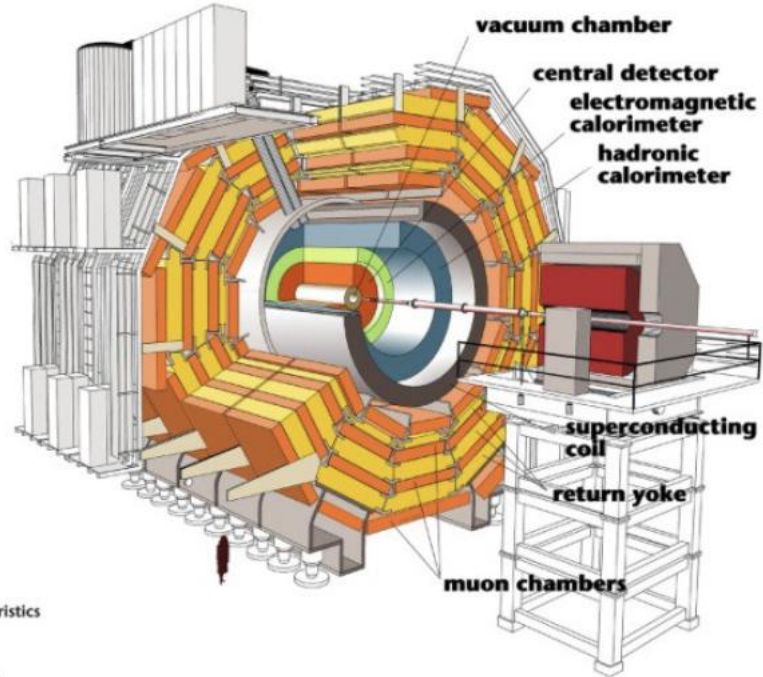
Fit Physicists
Ideas

Into Computing Resources

O RLY?

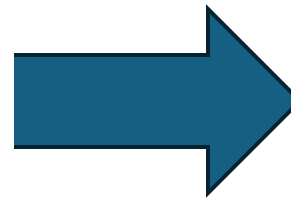
Harry Houdini

From RAW data to physics results

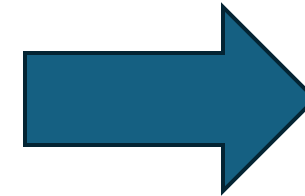


Detector characteristics

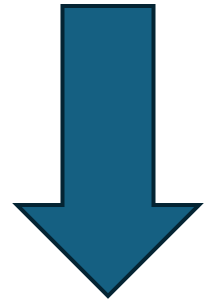
Width: 22m
Diameter: 15m
Weight: 14'500t



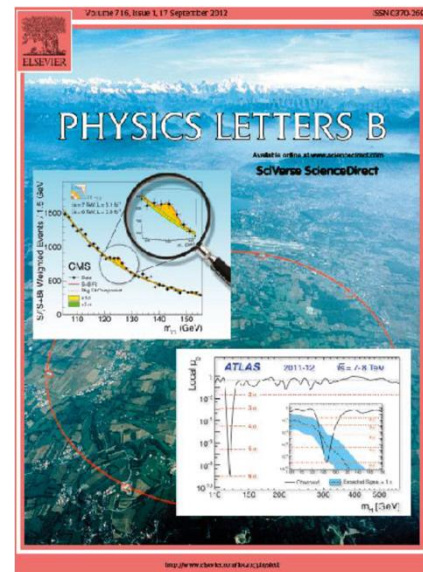
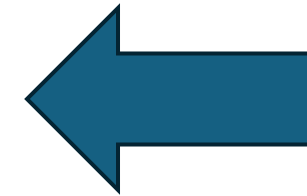
Trigger
DAQ



Data
preparation



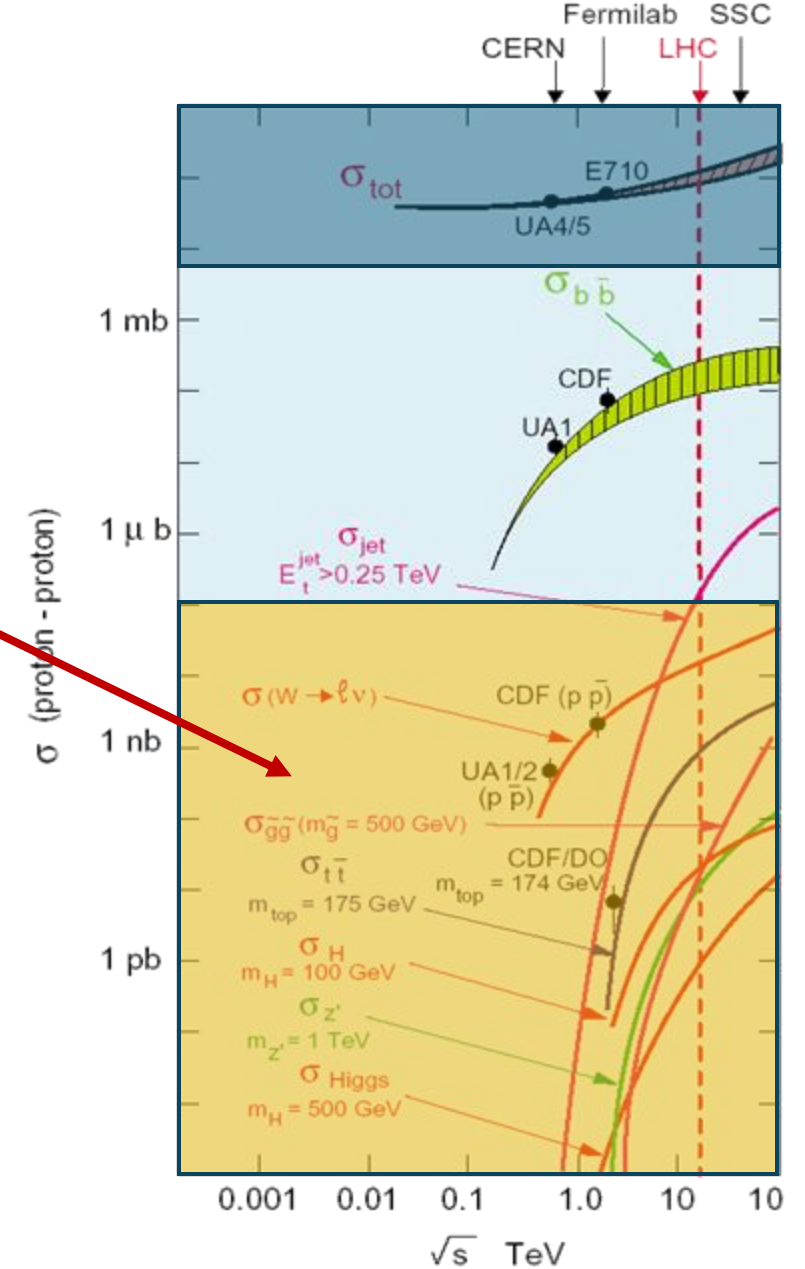
Analysis



Online processing

Trigger strategy to **select rare processes**:

- search for local signatures (calorimeter energy, presence of muons...)
- Reject background
- Select rare events



Online processing

Trigger strategy to **select rare processes**:

- search for local signatures (calorimeter energy, presence of muons...)
- Reject background
- Select rare events



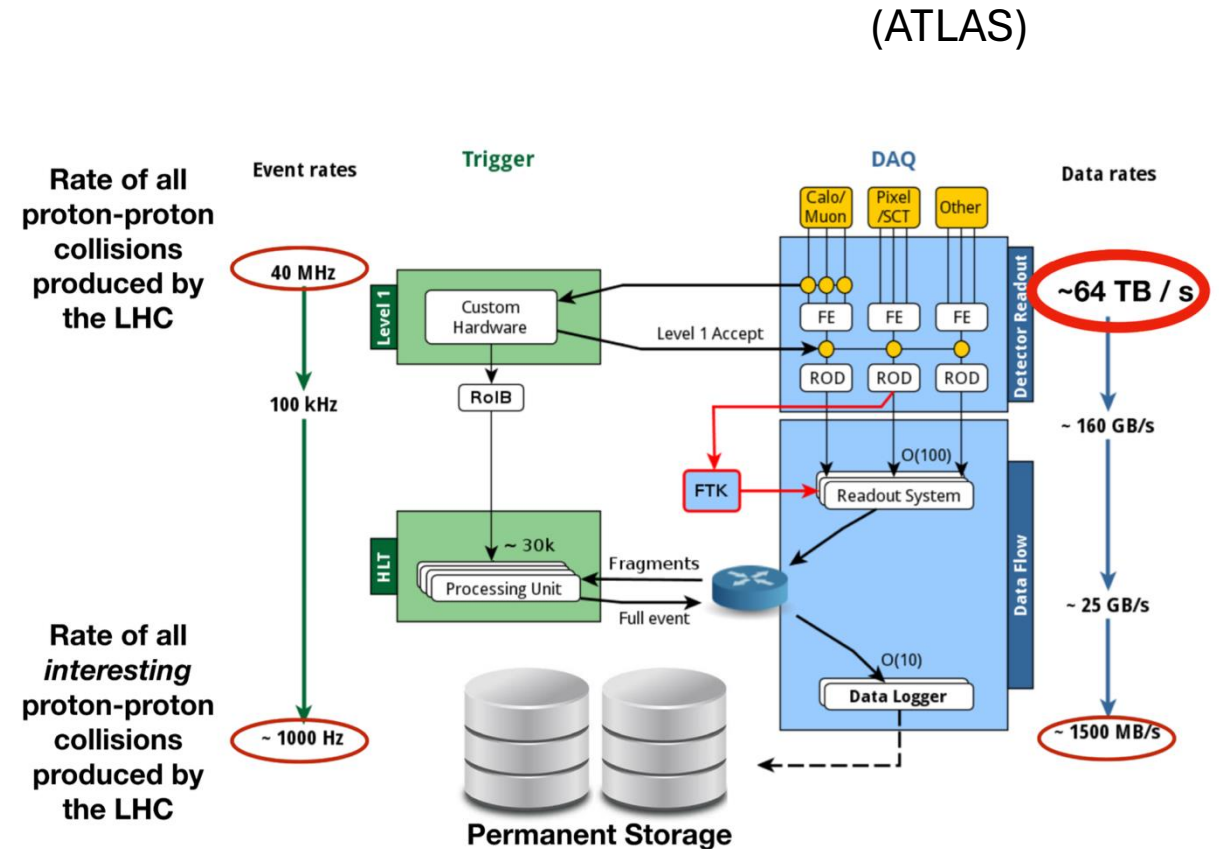
Online processing

Trigger strategy to **select rare processes**:

- search for local signatures (calorimeter energy, presence of muons...)
- Reject background
- Select rare events

“Classic” multi-level trigger

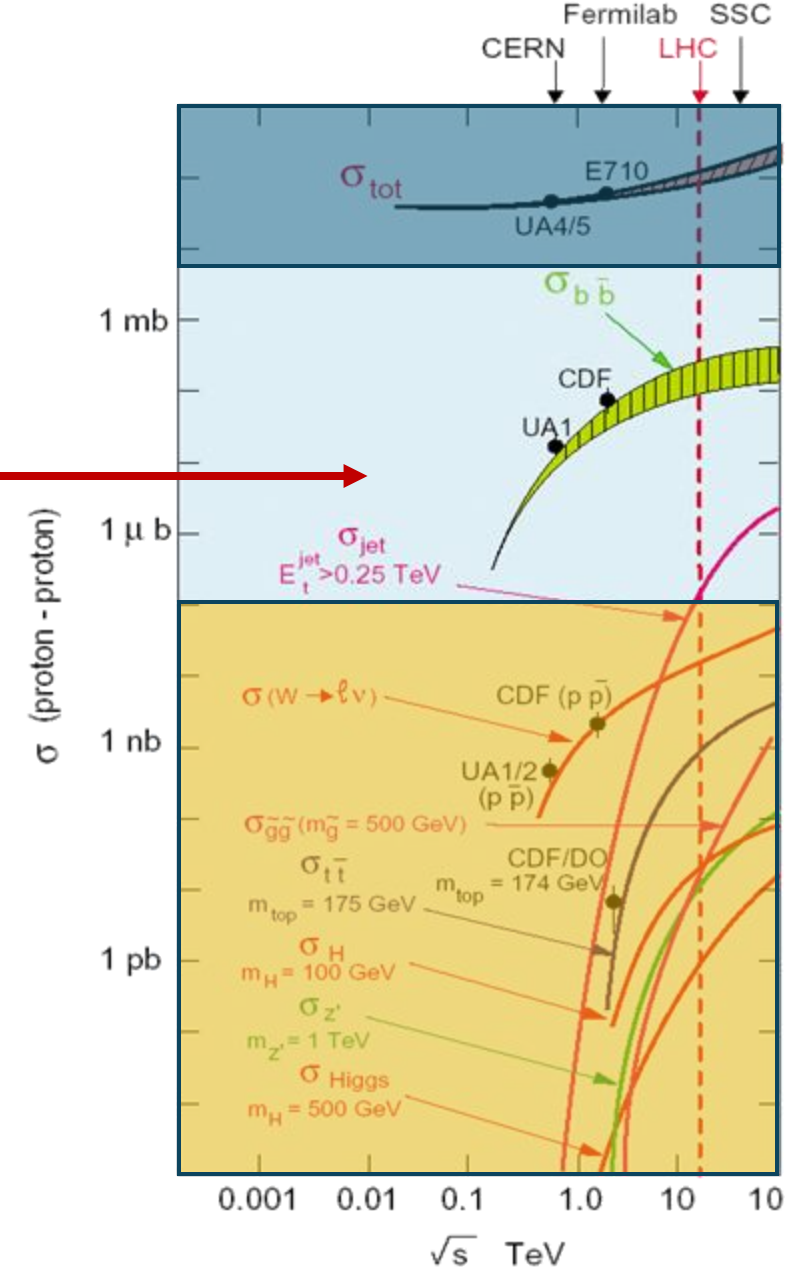
- Chain of “yes or no” decisions
- Very fast first level with (programmable) hardware
- “slower” higher level(s) via software on specialised or off-the-shelf processors



Online processing

Trigger strategy to select **high cross-section, signal-dominated processes**:

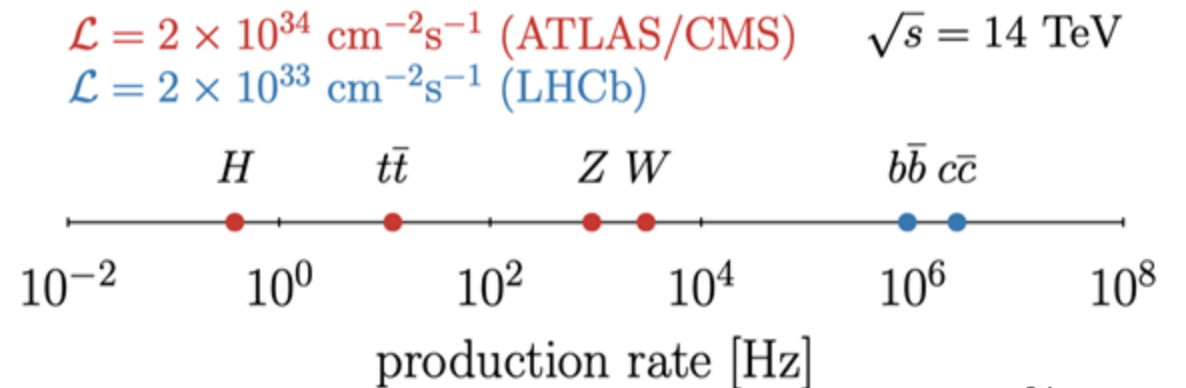
- No “simple” local criteria
- Classify decays
- Access as much information about the collision as early as possible
- Read full detector



Online processing

Trigger strategy to select **high cross-section, signal-dominated processes**:

- No “simple” local criteria
- Classify decays
- Access as much information about the collision as early as possible
- Read full detector

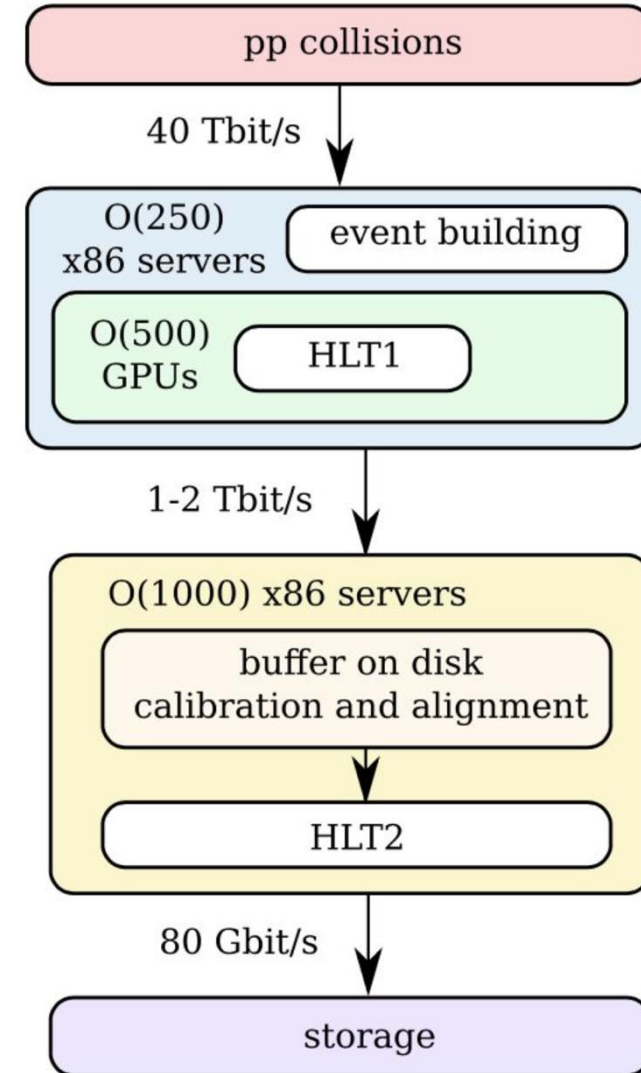


Online processing

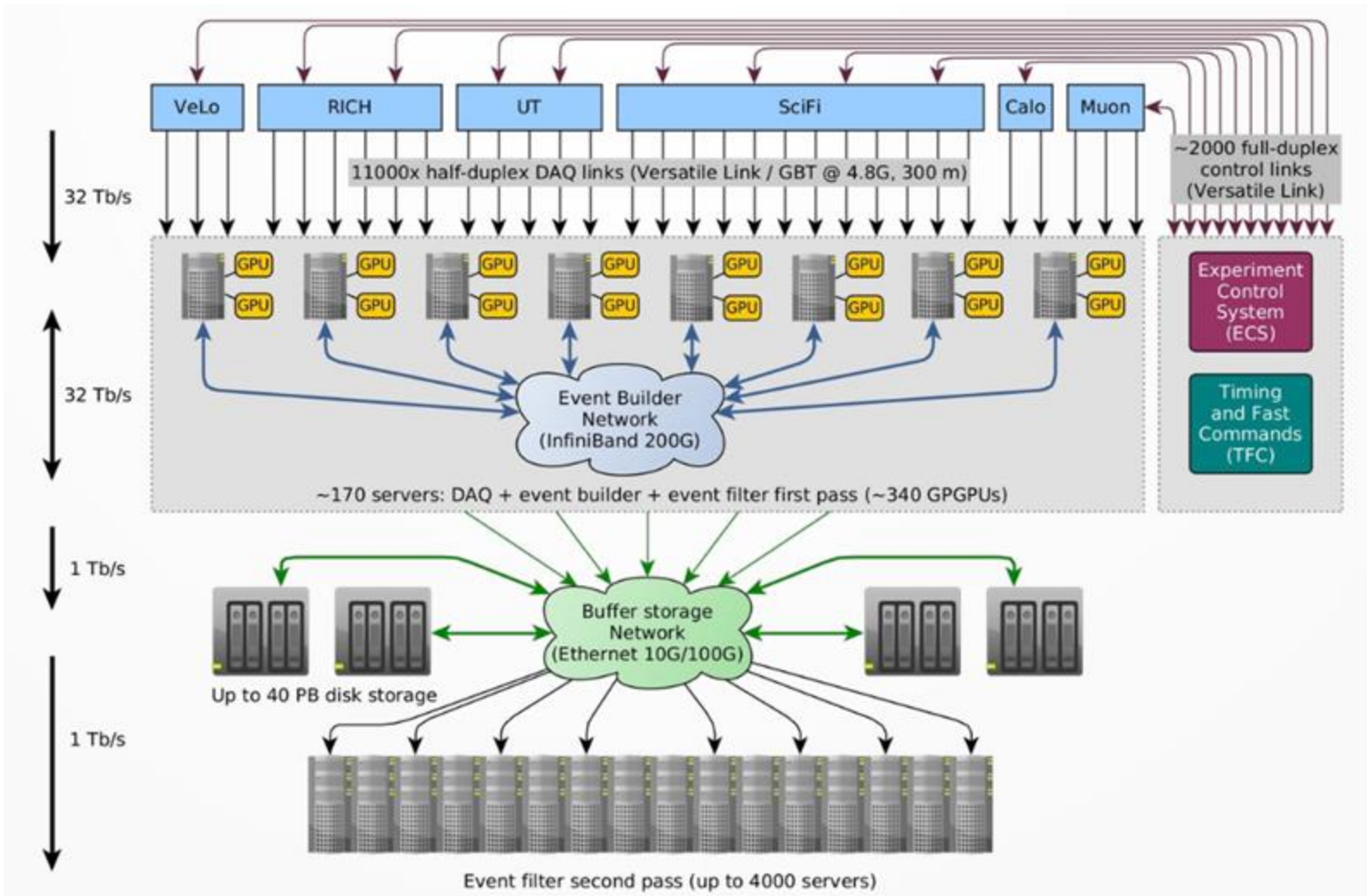
(LHCb)

Trigger strategy to select high cross-section, signal-dominated processes:

- No “simple” local criteria
- Classify decays
- Access as much information about the collision as early as possible
- Read full detector



Schematic view of LHCb DAQ & trigger system



~19000 fibre links between the detector (100m underground) and the computing farm

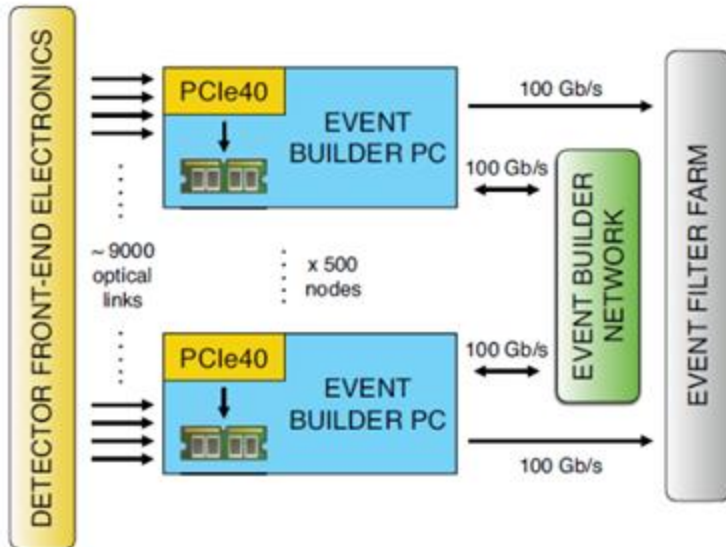
170 servers to read the data and pre-filter (HLT1, running on GPUs)

More than 4000 CPU servers for the event filter second pass

Up to 40PB of disk storage in between

Reading from multiple channels, with synchronization needed!!!

LHCb Data Acquisition



- Event rate: 30 MHz non-empty bunch crossing
- Event size: ~ 100 kB
- Input bandwidth: 40 Tbit/s



©CPPM

- New PCIe40 readout boards
 - 24 optical inputs, PCIe interface
- Event builder network using commercial technology
 - HDR InfiniBand© with remote direct memory access

Uses Field Programmable Gate Arrays (FPGA) for logic (programming with VHDL or Verilog)

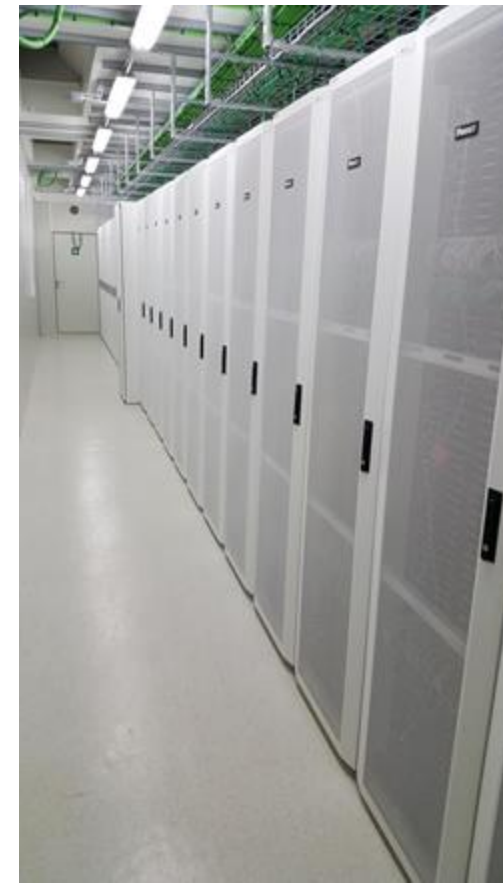
LHCb Online farm



LHCb
computing
center at CERN



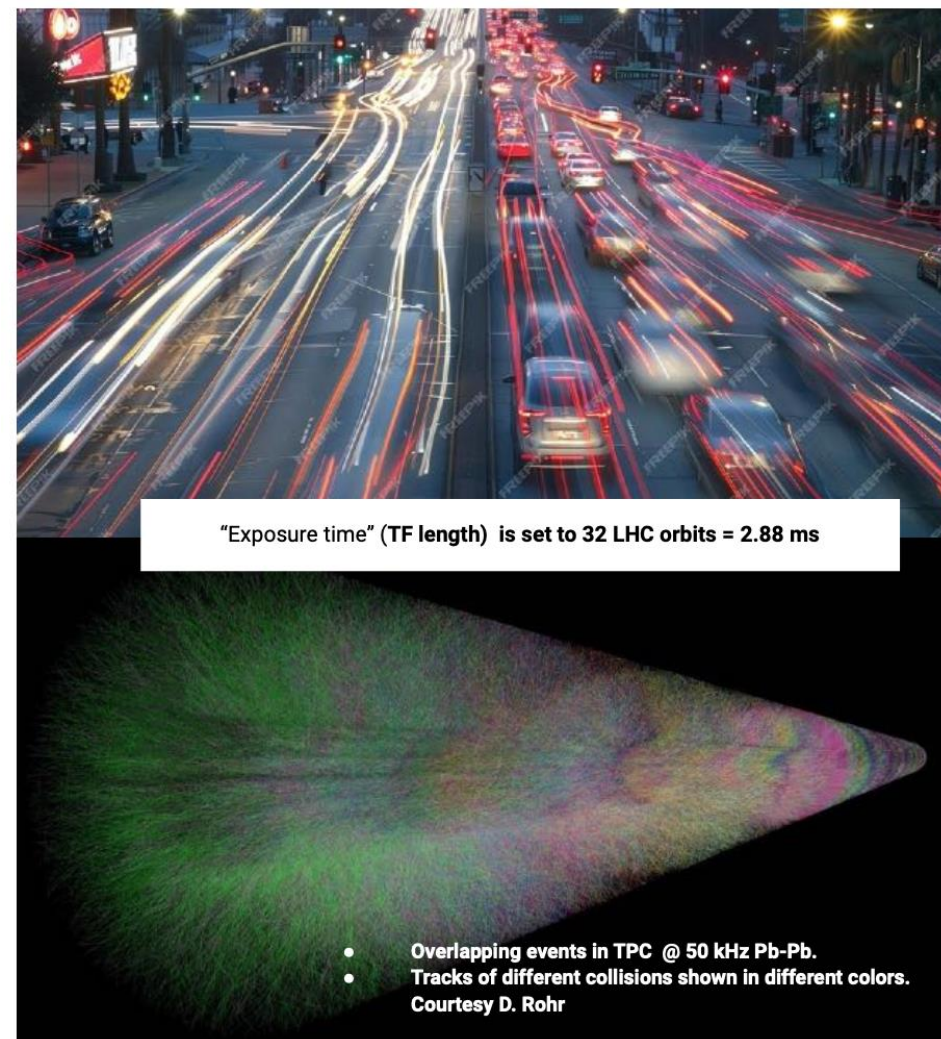
19000 fibres connect the experiment
with the computer center



More than 4000 servers in the racks

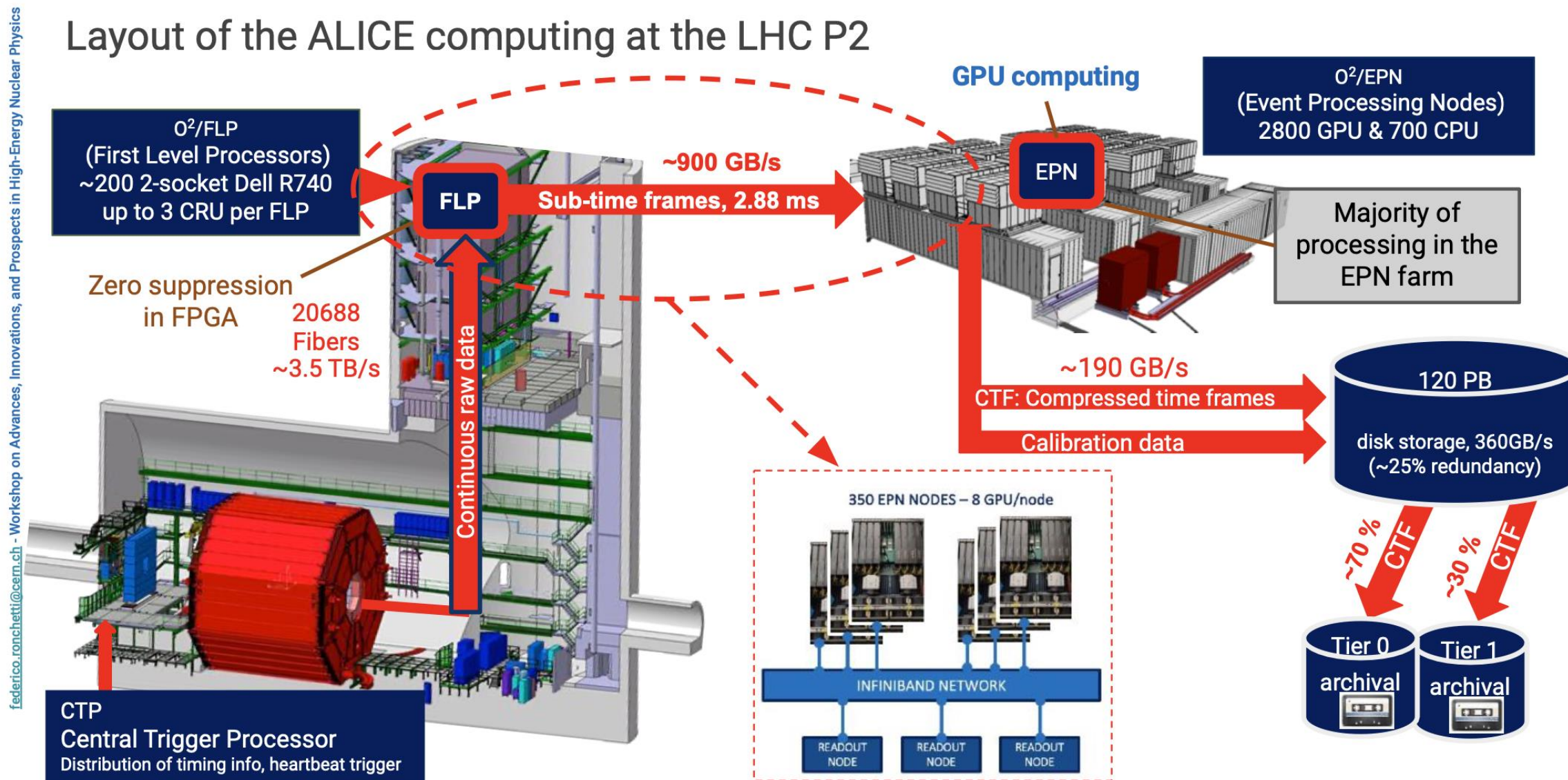
The Run3 ALICE computing

- The **Time Frame (TF)** concept
 - All collisions stored for main detectors
→ no trigger
 - Collection of tracks in a given time window (“long exposure photograph”)
 - Exposure time tunable (~ 2.88 ms)
 - 100x more collisions
 - Raw TF stream input to GPU farm where tracking, reconstruction and compression are performed in sync with data taking
 - Storing compressed TFs
- Online (sync) and offline (async) reconstruction using exactly the same code

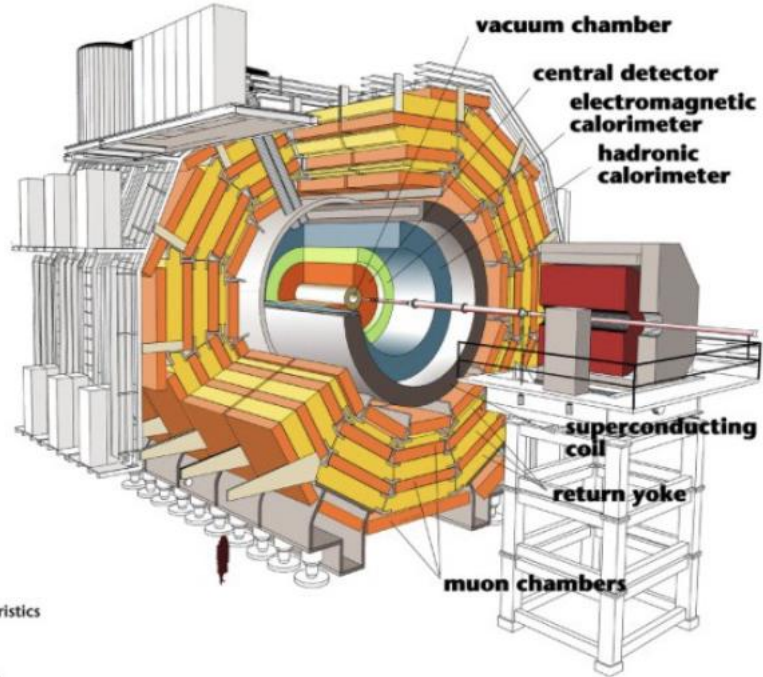


The ALICE Run3 data flow

Layout of the ALICE computing at the LHC P2

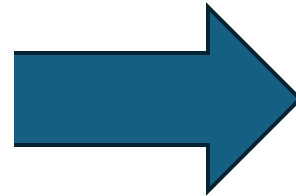


From RAW data to physics results

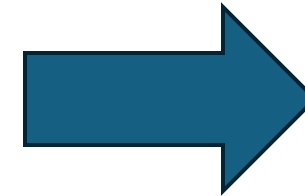


Detector characteristics

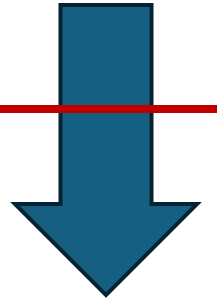
Width: 22m
Diameter: 15m
Weight: 14'500t



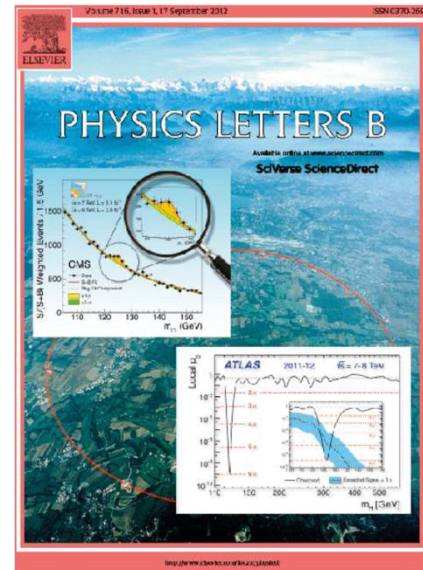
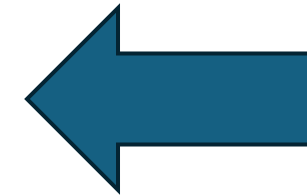
Trigger
DAQ



Data
preparation



Analysis



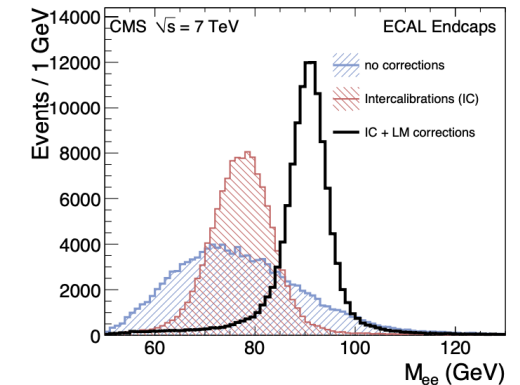
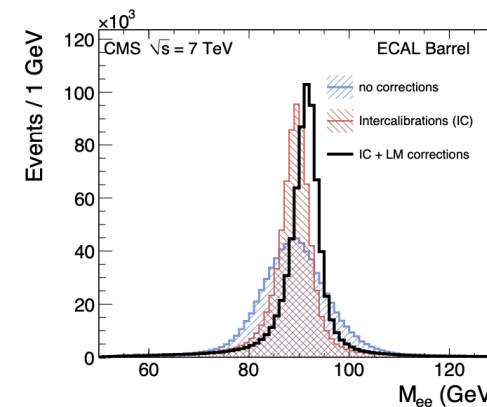
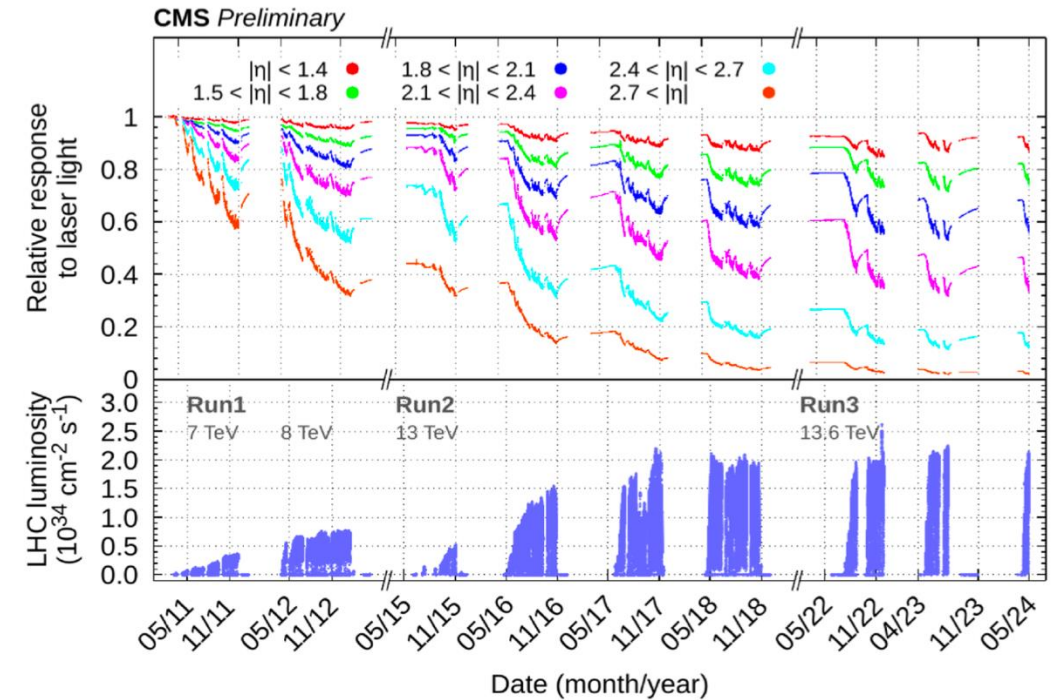
Data preparation

- **Interpretation of RAW detector signals into physics objects**
- **Calibration**
 - Convert raw data to physical quantities
- **Alignment**
 - Find out precise detector positions
- **Event reconstruction**
 - Reconstruct particle tracks and vertices (interaction points)
 - Identify particle types and decays
 - Impose physics constraints (energy and momentum conservation)
- Used to happen offline; trend to move online parts or even the entire chain

Calibration

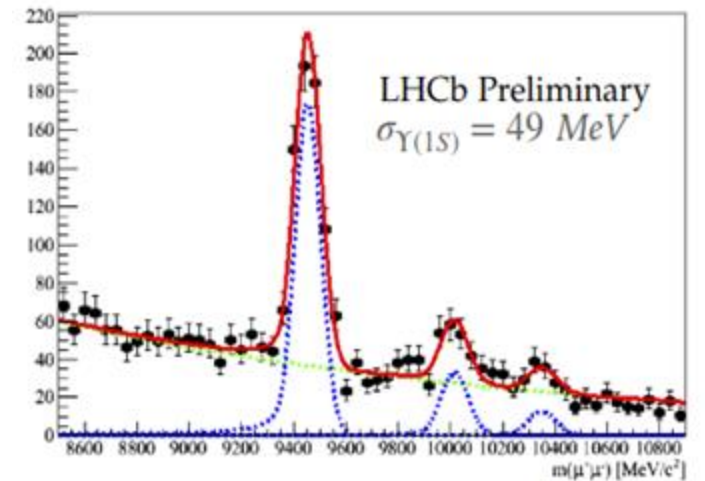
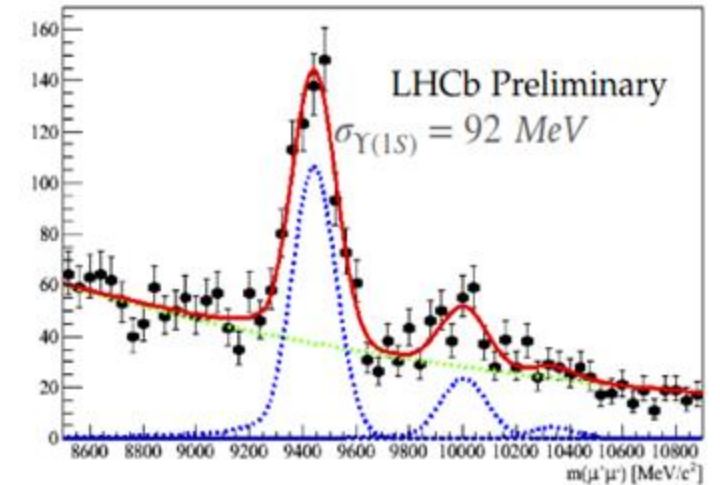
- Raw data are mostly ADC or TDC counts
- They have to be converted to physical quantities such as energy or position
- Very detector-dependent
- Every detector needs calibration
- Calibration constants need to be updated and stored in a database

(CMS ECAL)



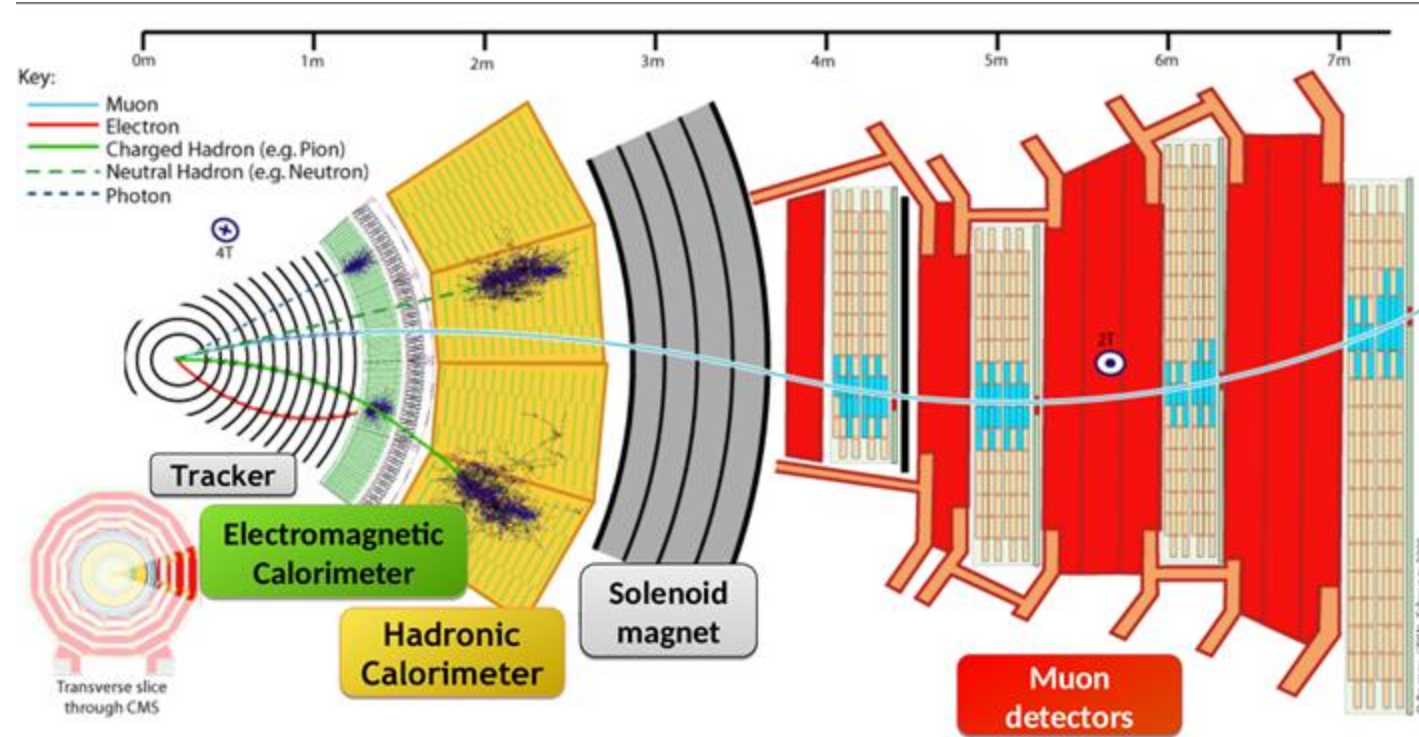
Alignment

- Tracking detectors are very precise instruments
 - Silicon strip detector: $\sim 50\ \mu\text{m}$
 - Pixel detector: $\sim 10\ \mu\text{m}$
 - Drift tube: $\sim 100\ \mu\text{m}$
- Positions of detector elements need to be known to a similar or better precision
- Alignment with charged tracks from collisions, beam halo and cosmic rays
 - Continuous process
 - Alignment constants need to be updated and stored in a database



Reconstruction

- Find out which particles have been created where and with which momentum
- Reconstruct charged particles
- Reconstruct neutral particles
- Identify type of particles
- Reconstruct vertices (interaction points)
- Reconstruct kinematics of the interaction
- Not trivial, very time-consuming ...

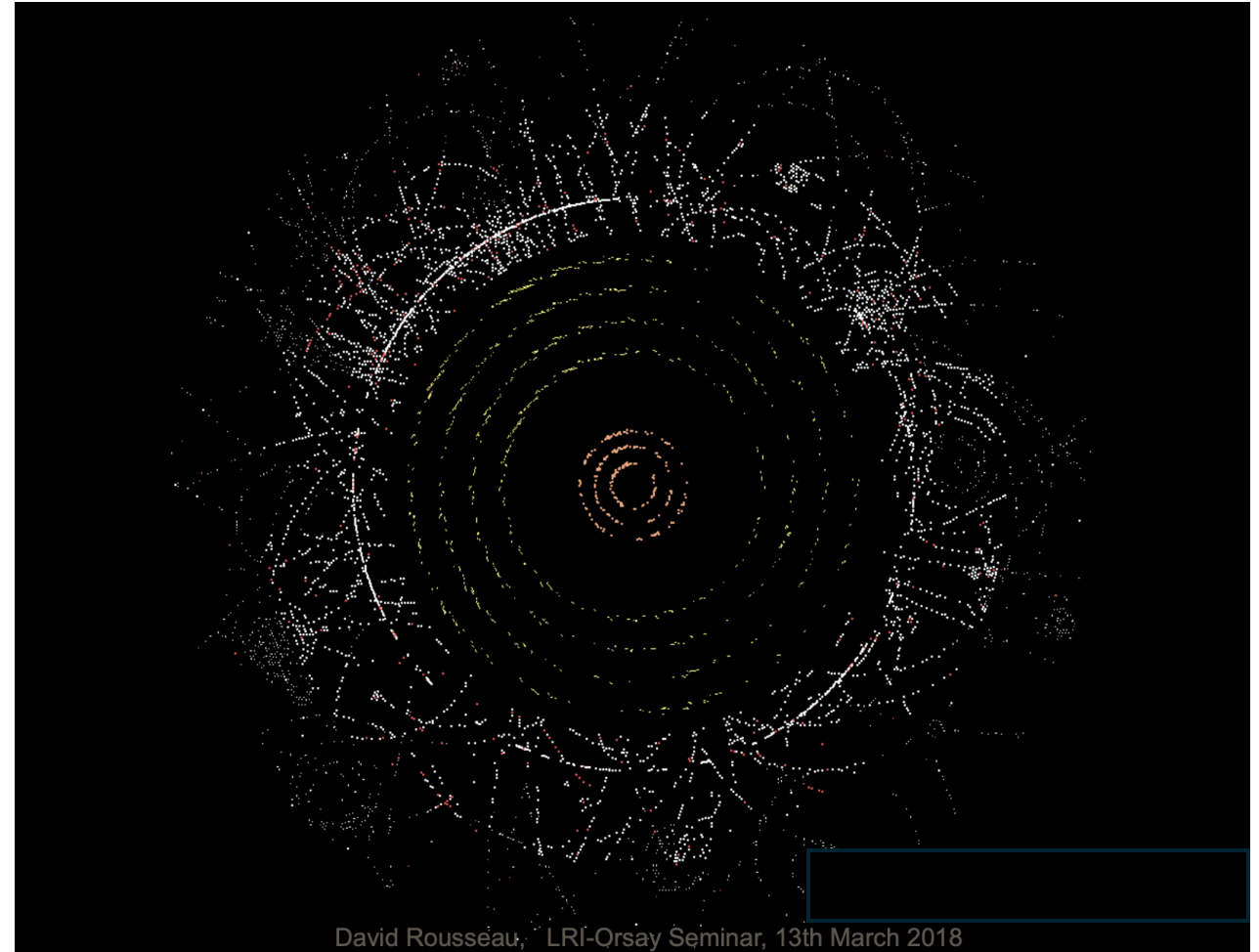


There are many many types of sensors, and other types of detectors (e.g. the Time Projection Chamber in ALICE)

Very active field, with uses outside of High Energy Physics

Reconstruction of charged particles

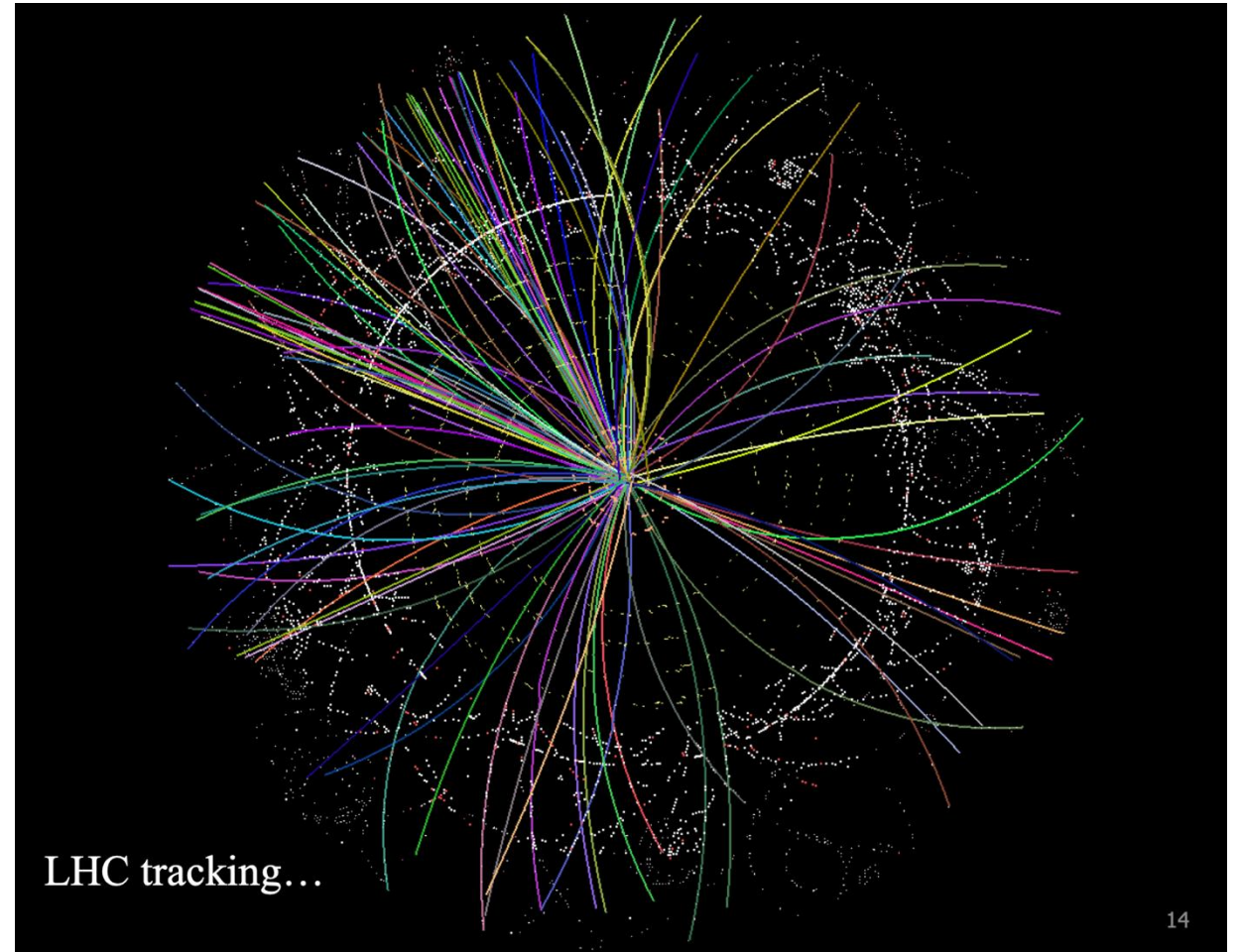
- Measure position in detector layers (“hits”)
- Curved trajectory due to the magnetic field
- Use position measurements to determine track parameters (location, direction, momentum) and their uncertainties



David Rousseau, LRI-Orsay Seminar, 13th March 2018

Reconstruction of charged particles

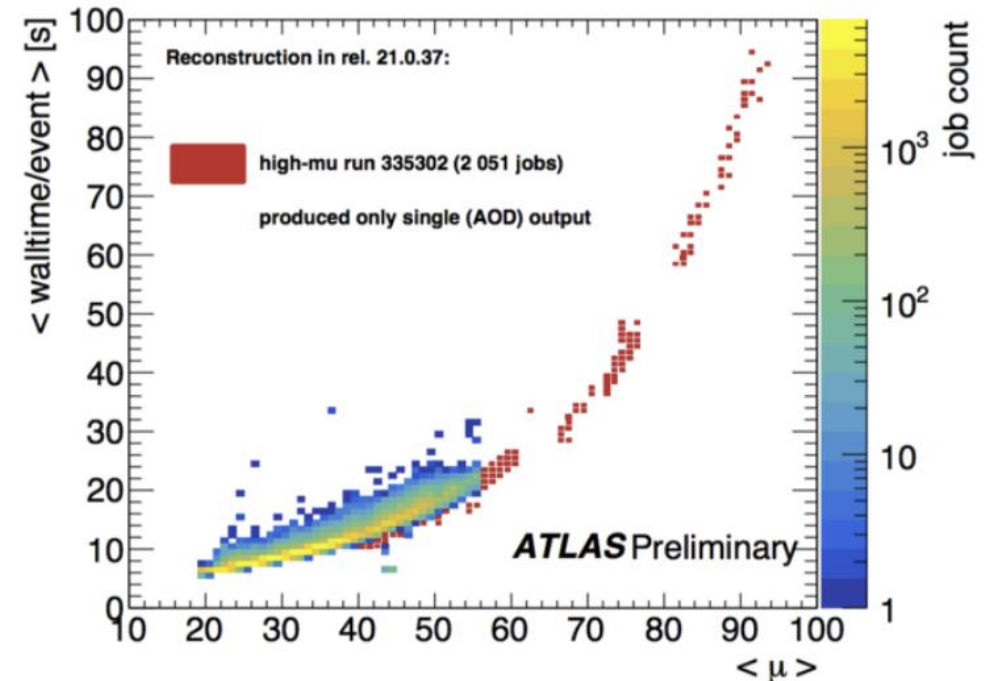
- Measure position in detector layers (“hits”)
- Curved trajectory due to the magnetic field
- Use position measurements to determine track parameters (location, direction, momentum) and their uncertainties



Reconstruction of charged particles

Difficult task!

- Assignment of hits to particles is unknown
- Huge background from low-momentum tracks
- Material interactions: Multiple Coulomb scattering, Energy loss (ionization, bremsstrahlung)
- Mathematically complex (Kalman Filter, matrix algebra, propagation in a not uniform magnetic field)
- Highly combinatorial: given a set of N signals, it scales as N^M , with $M > 1$ and algorithm-dependent



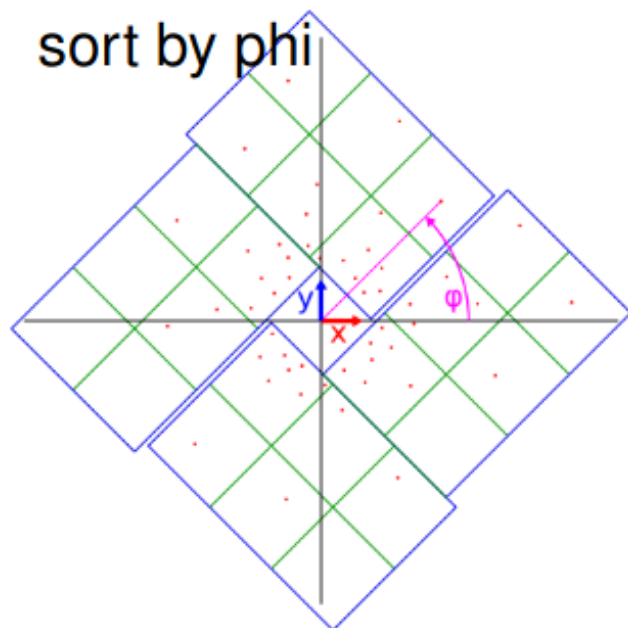
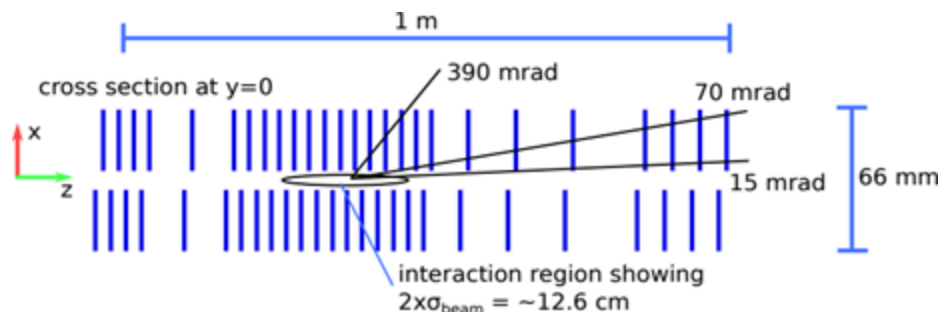
Problem decomposition

- **Pattern Recognition or Track Finding**
 - Assign detector hits to track candidates (collection of hits all believed to be created by the same particle)
- **Parameter estimation or Track Fit**
 - Determine track parameters + their estimated uncertainties (covariance matrix)
- **Test of the track hypothesis**
 - Is the track candidate the trace of a real particle?

Track finding

- Very detector-dependent
- Many solutions available, no general recipe
- Global methods: include all measurements (clusters) in a formulation of the problem where solutions match to tracks
- Local methods: iterative methods where a seed is found first and is there forwarded to other sensors

A local method: Search by triplet in the LHCb VeLo



Back view of the Velo
(The beam travels on the Z axis)

Clusters from different sensors are grouped in tracks that crossed the detector

We can make hypotheses to simplify the problem: our tracks come from the beam interaction point, this means that they stay at constant ϕ angle

The tracks are straight in the LHCb Velo as there is no magnetic field there

<https://arxiv.org/pdf/2207.03936>

A local method: Search by triplet in the LHCb VeLo

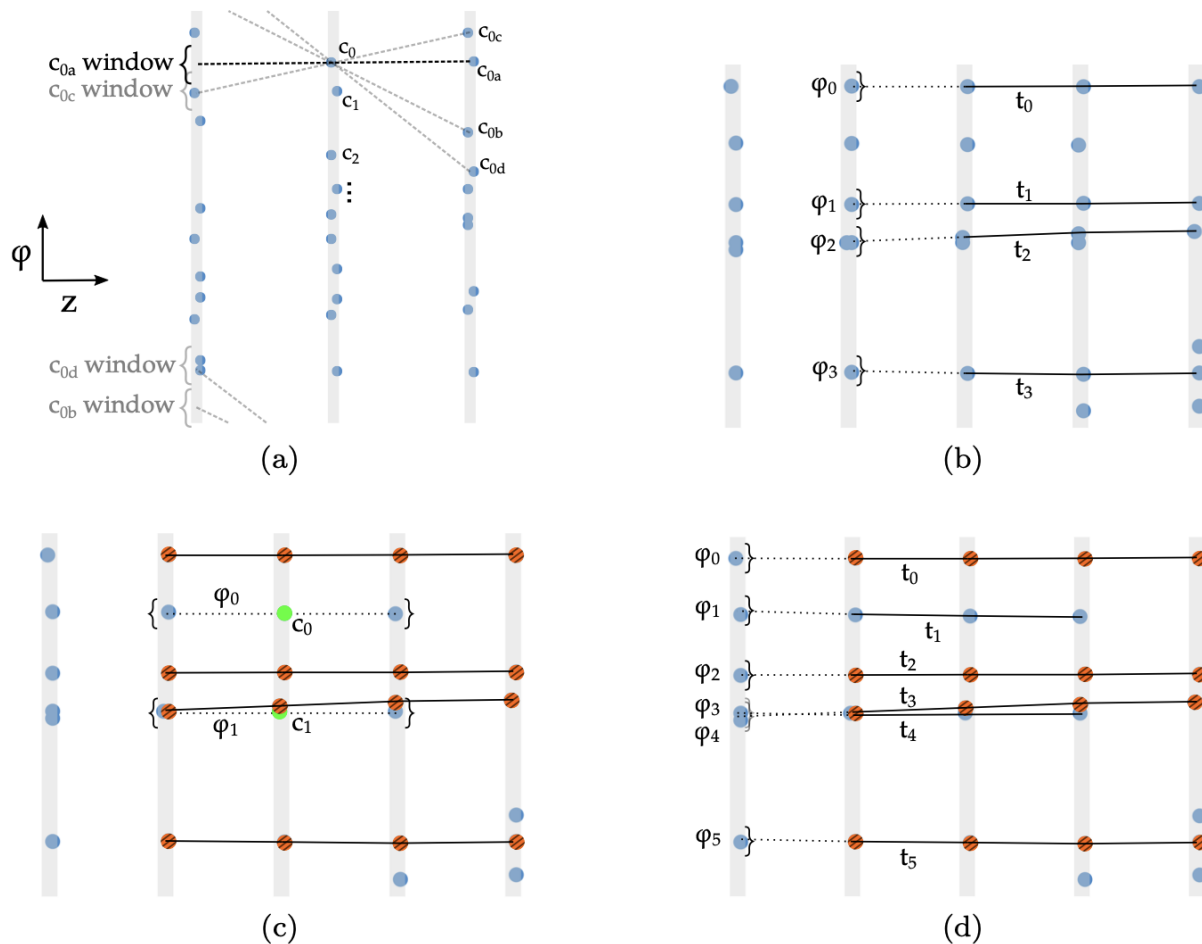
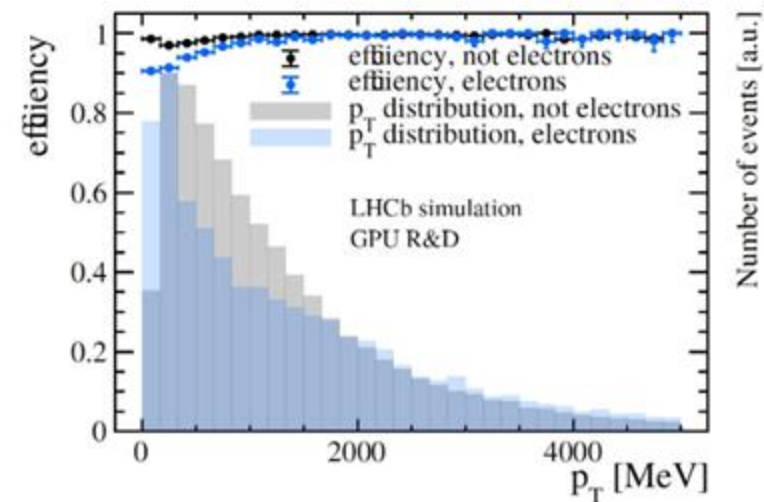


Figure 5: Iterative seeding and following stages, where modules are considered from right to left. (a) Seeding stage. For hit c_0 , four hits c_{0a} , c_{0b} , c_{0c} and c_{0d} are considered on the neighbouring module on the right. Each of the resulting *doublets* is extrapolated onto the neighbouring module on the left, where hits in the φ window are considered. The φ search wraps around. (b) Following stage. Forming tracks are extrapolated and hits are sought in a φ window. (c) and (d) Subsequent seeding and following stages. Hits found in previous follow stages are marked as flagged and not further considered.

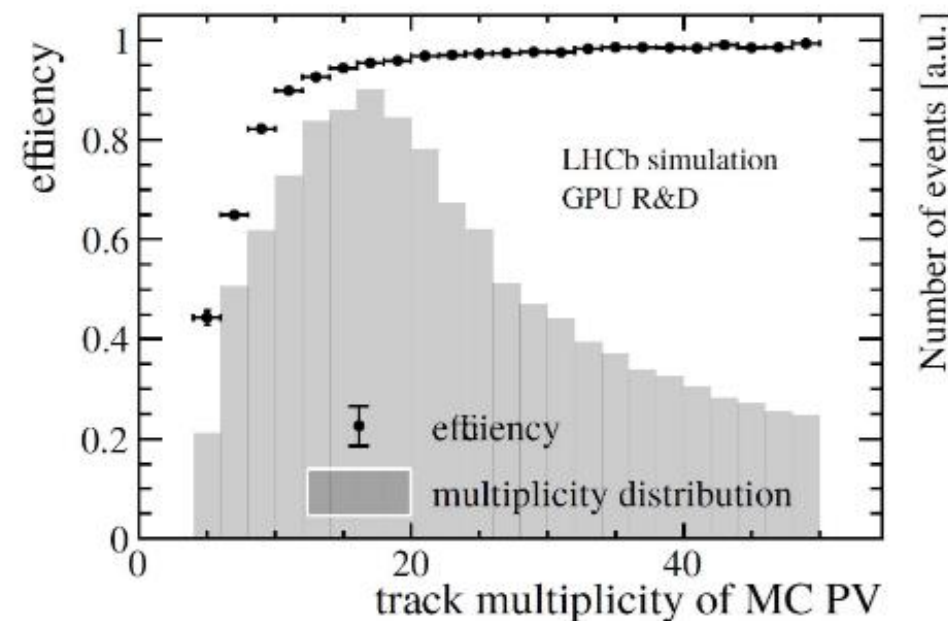
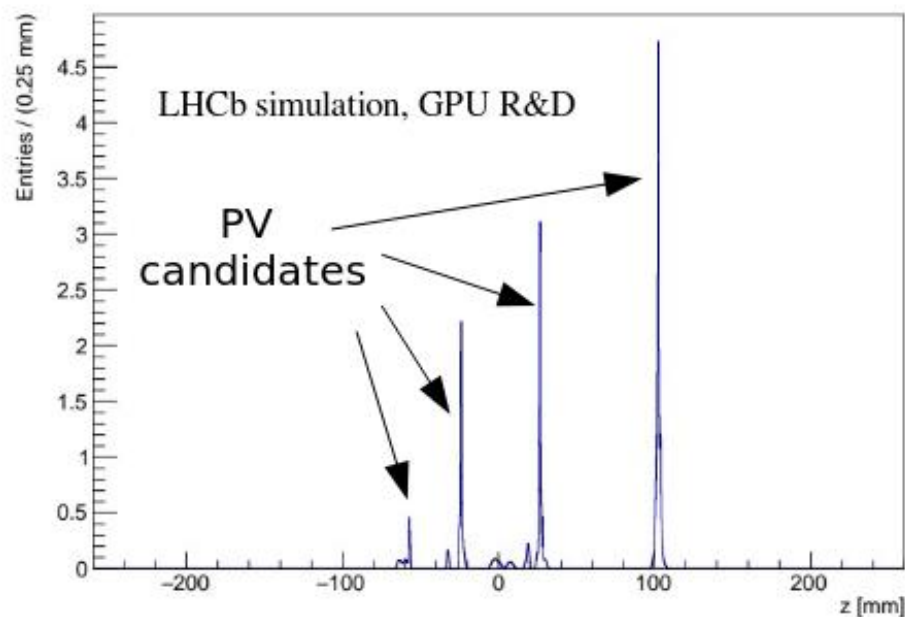
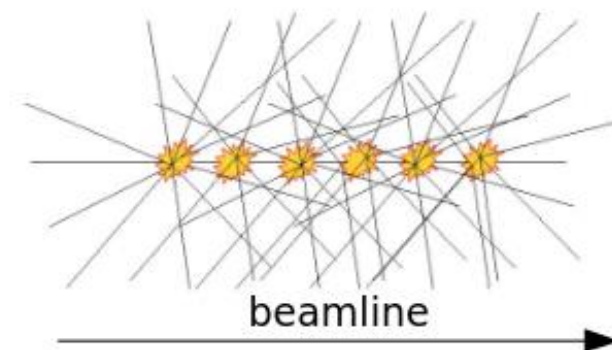


<https://arxiv.org/pdf/2207.03936>

Finding the primary collisions: Vertexing

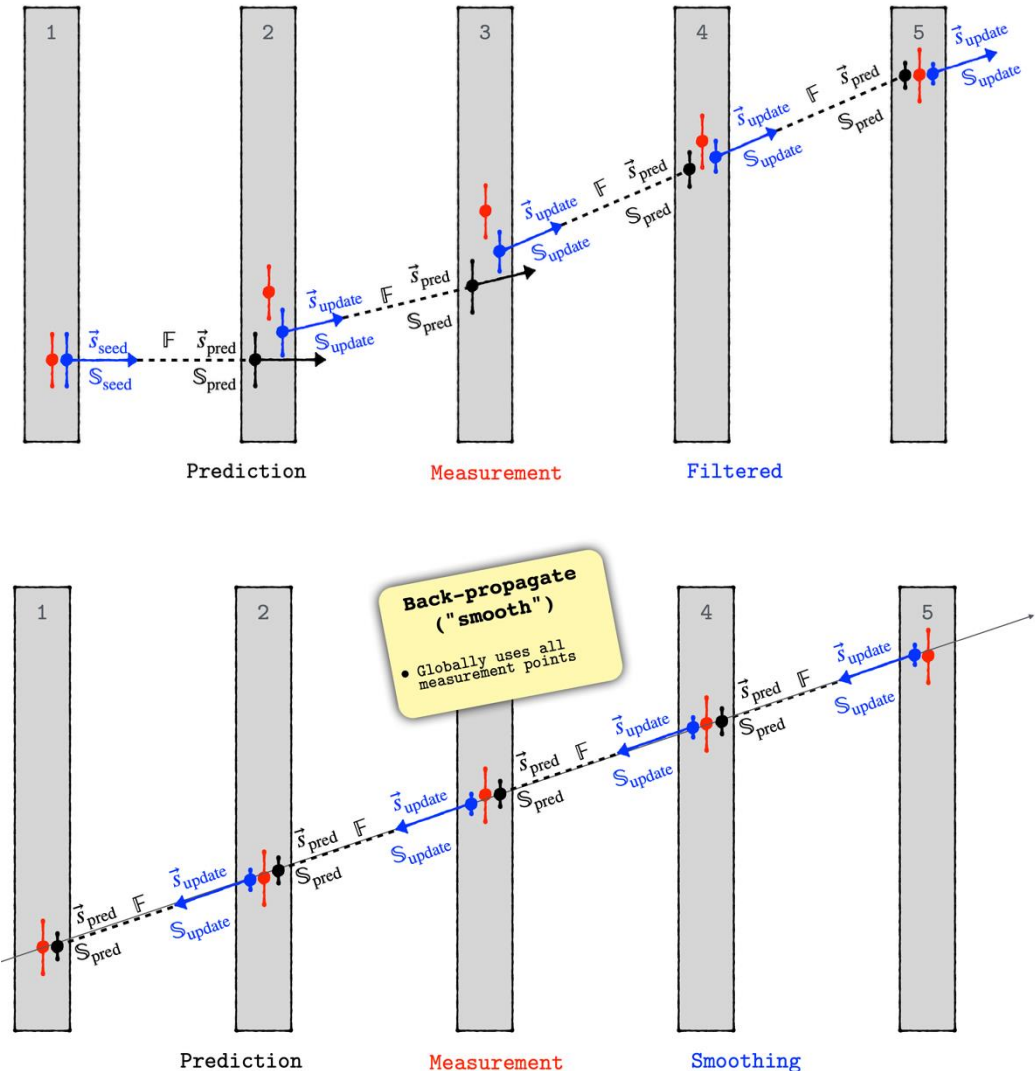
Record z of closest approach to beamline for each track

Peaks in distribution identify PVs

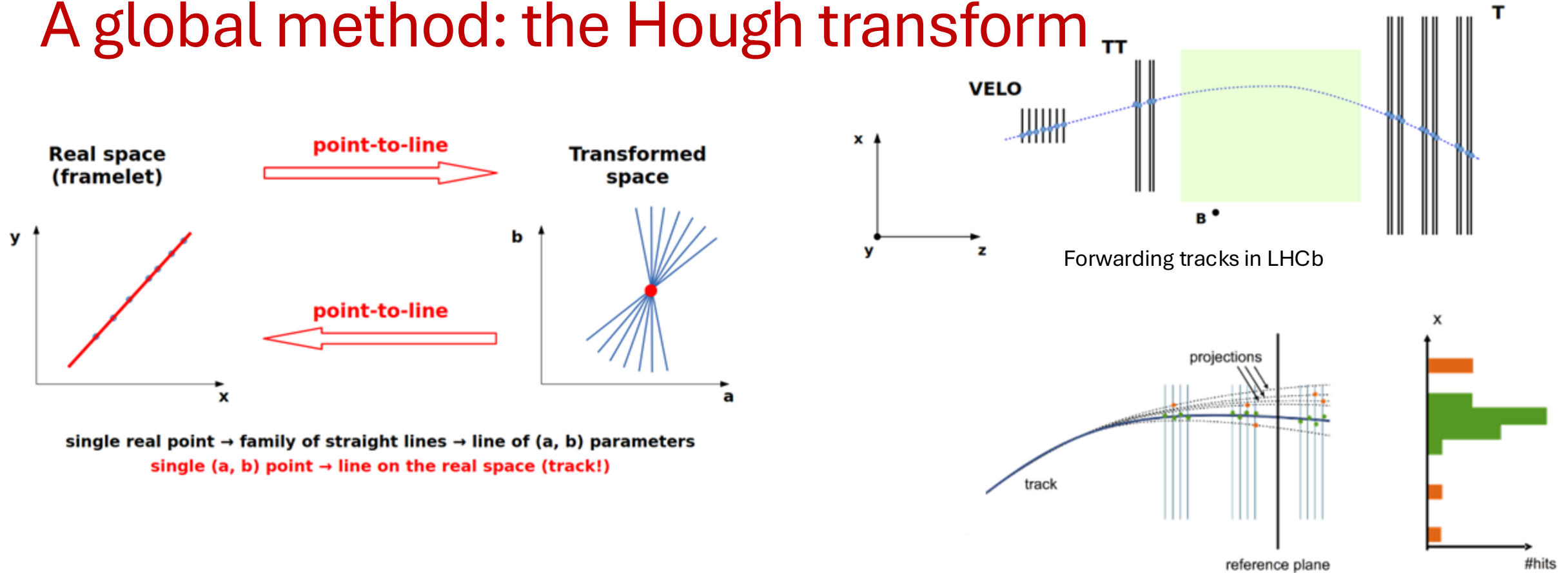


Combinatorial Kalman filter

- A “seed” (initial measurement and covariance) defines the initial state and its covariance
- A propagation operator defines the predicted state on the following layer
 - Noise (e.g. multiple scattering) taken into account
- State is updated with measurement information, and propagated to the next layer
- ...and so on, until the last layer
- Back propagation “smooths” the track parameters by globally using all measurement points
- Wrong combinations of hit associations (fake tracks) are reduced by starting from a very pure track seeds (e.g. pixel triplets)



A global method: the Hough transform

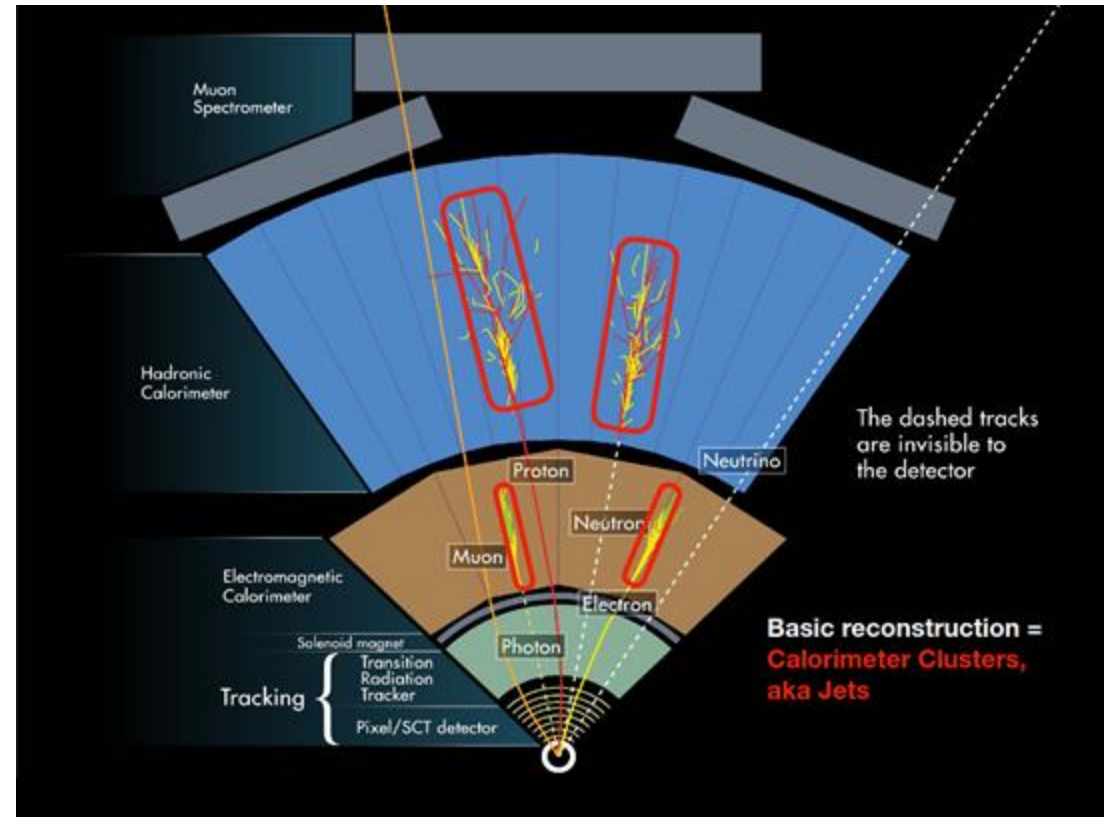


Map each point to a line in the space of parameters (a and b are the parameters defining the line), and look for accumulation points in the transformed space.

Can be **Computationally expensive**, but can be optimized and is useful in e.g. LHCb to forward Velo tracks downstream.

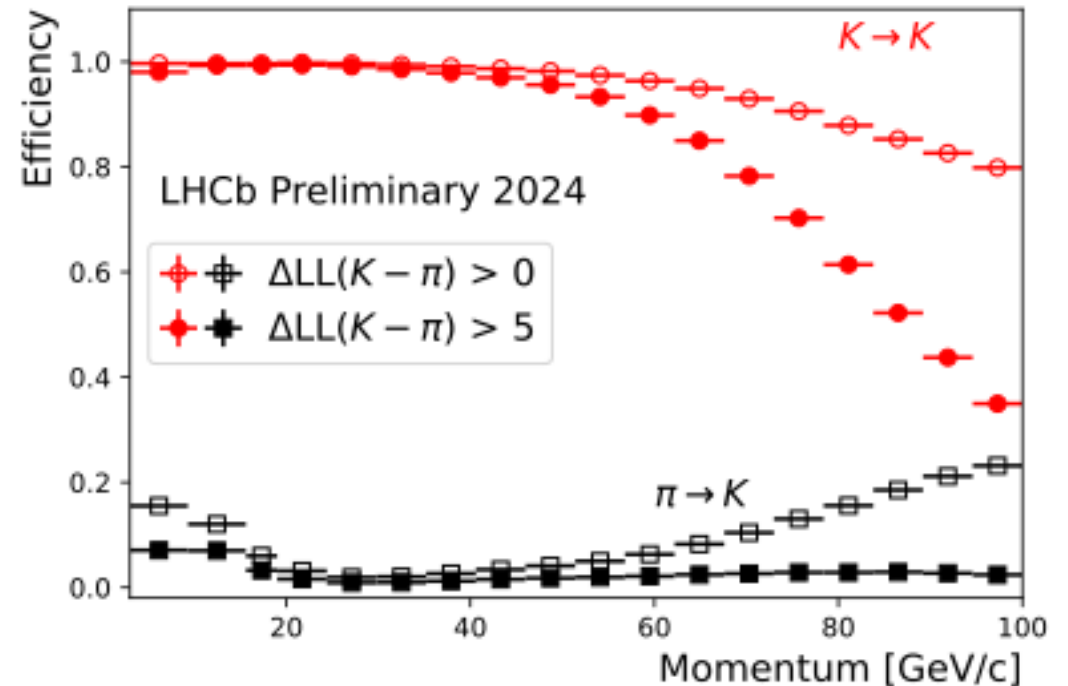
Neutral particles

- An incident neutral particle produces a shower in the calorimeter
 - cluster of cells with energy deposit above threshold
 - overlapping clusters must be separated
 - The cell-to-cluster association is a pattern recognition problem
- Various clustering techniques are used to find showers
- The algorithms depend on various characteristics of the calorimeter
 - Type (electromagnetic or hadronic)
 - Technology (homogeneous or sampling)
 - Cell geometry, granularity



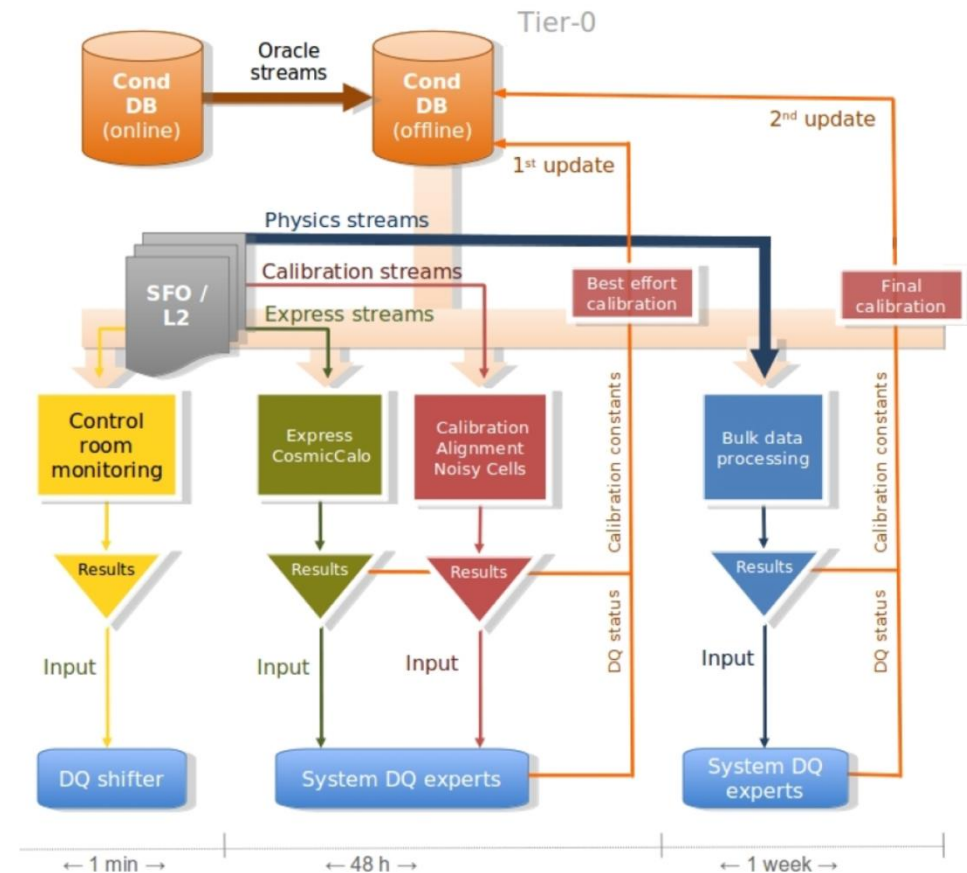
Particle Identification

- Dedicated detectors
 - Calorimeters
 - Cherenkov-based
 - Transition radiation
 - Time-of-flight
 - Ionization
- Information combined from several detectors
 - Using log-likelihoods, machine learning, etc.
 - Performance is monitored on data control samples

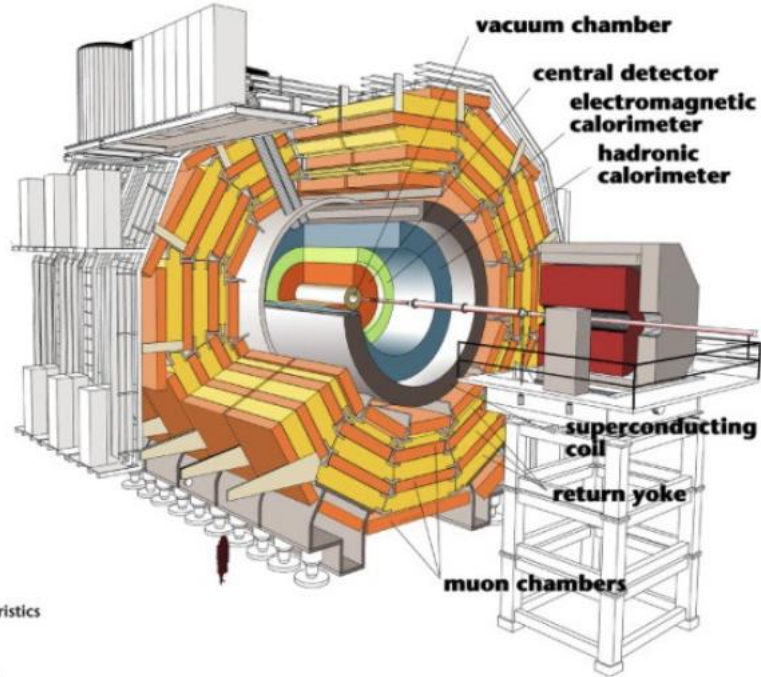


Data quality

- Only reliable, high-quality data are used
- The amount of useful data is maximised in data quality checks at several stages e.g.
 - In real-time during data-taking
 - after a quick calibration/alignment on a fraction of data
 - With the best calibration/alignment on all data

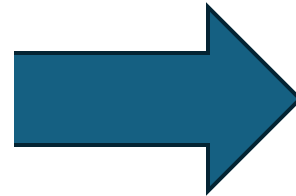


From RAW data to physics results

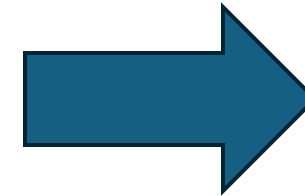


Detector characteristics

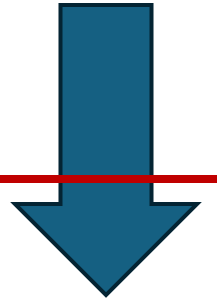
Width: 22m
Diameter: 15m
Weight: 14'500t



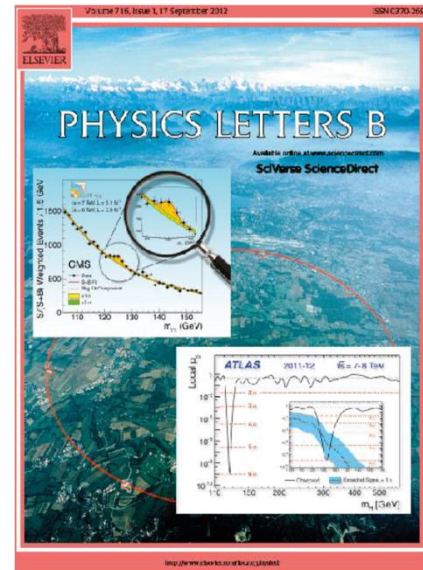
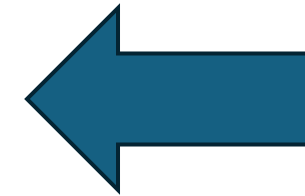
Trigger
DAQ



Data
preparation



Analysis



Physics analysis

- Extract physics signals from background
- Measurement or discovery limits of masses, cross-section, branching fractions, and other physics observables
- Further selection of events, with a statistical interpretation
- high dimension likelihoods on million / billions of events, use of sophisticated multi-variate techniques

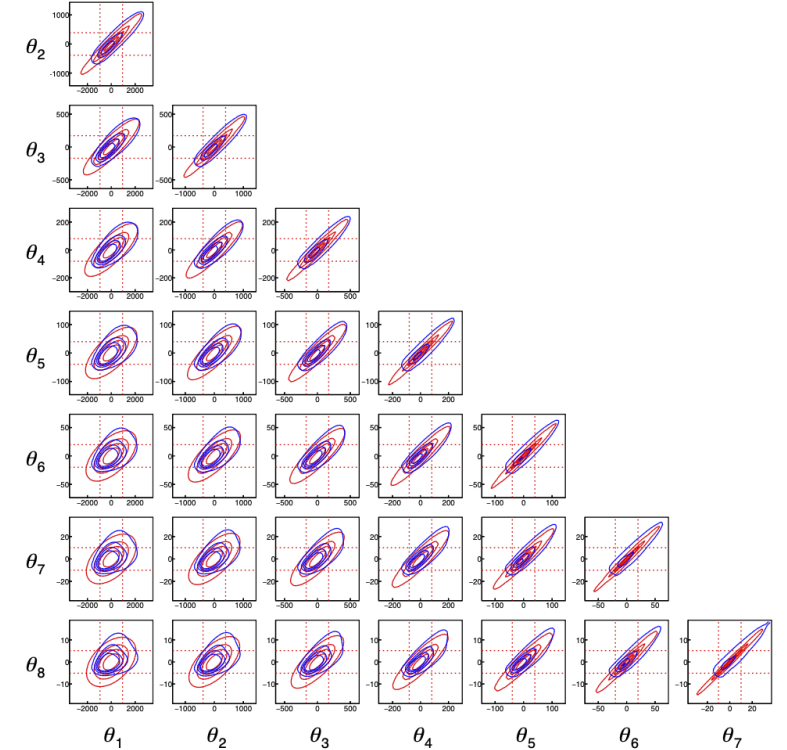
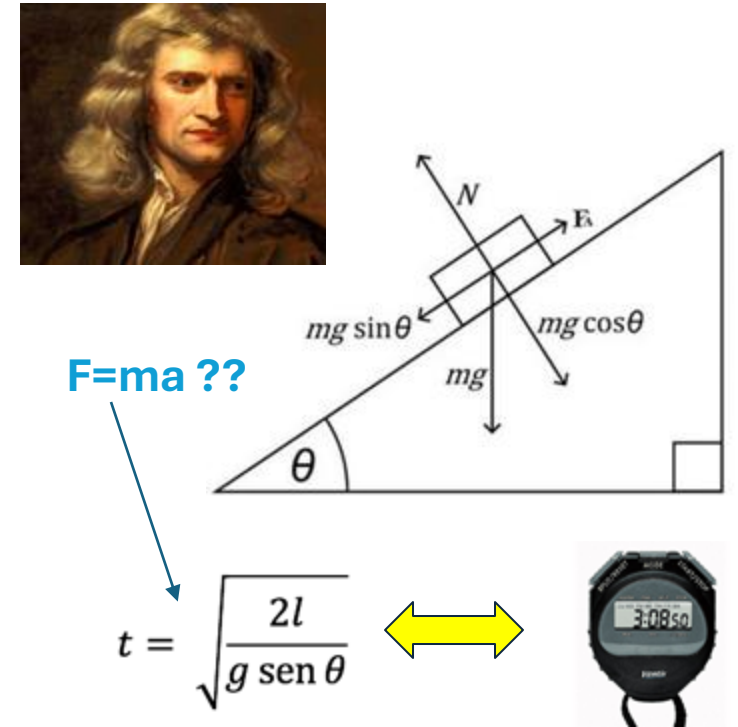
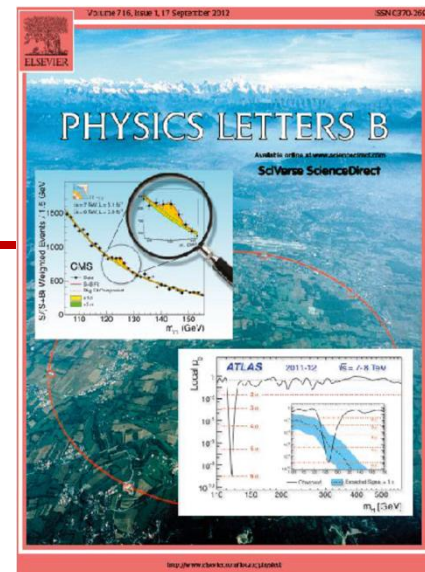
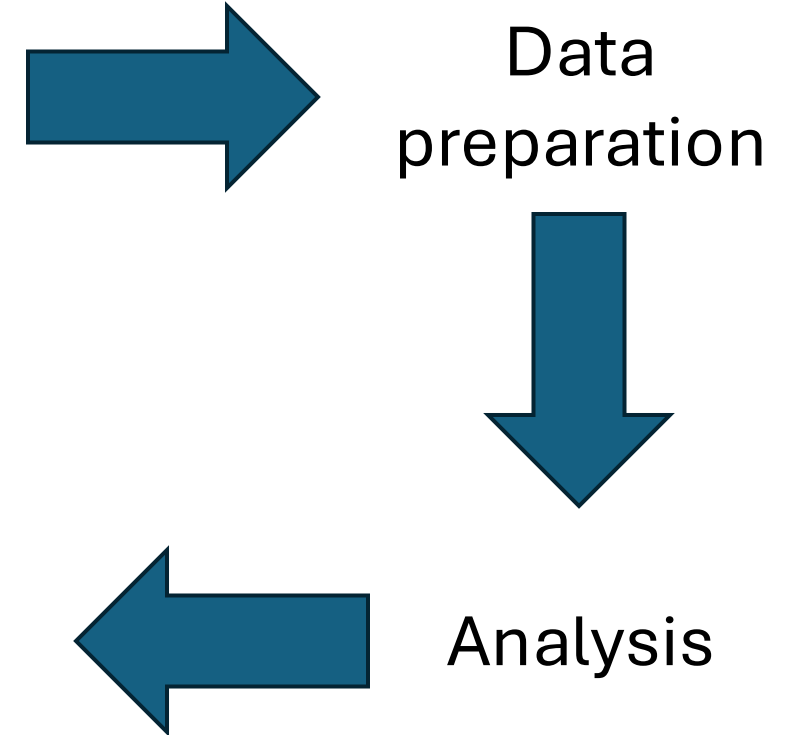
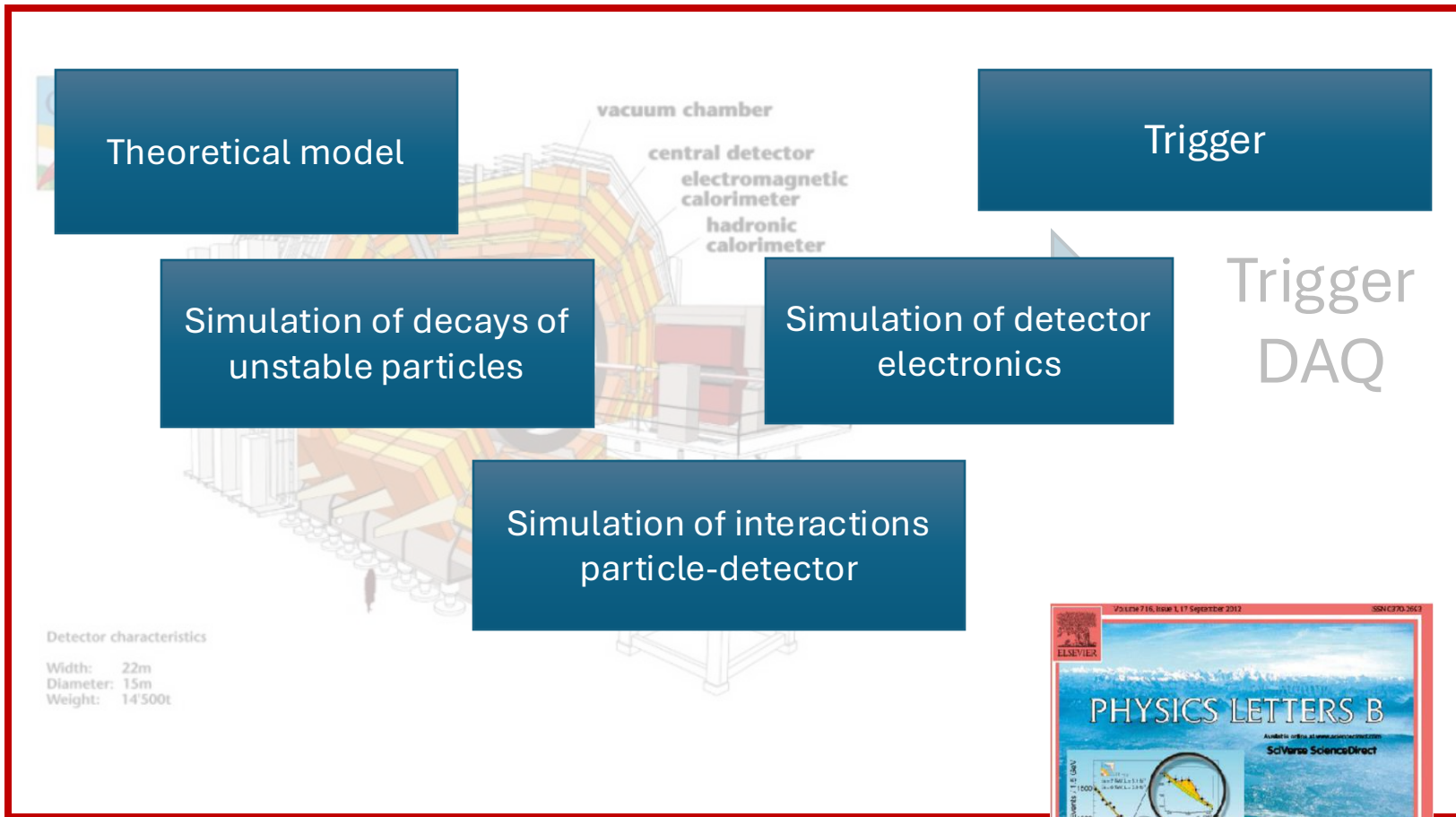


Figure 9: Contours of constant probability density for the true probability density function and the Gaussian approximation for the nuisance parameters in the toy search where an asymmetric background systematic is included. The red dotted horizontal and vertical lines indicate the regions for which $|\theta_i| < \sqrt{V_{ii}}$, where θ_i is the nuisance parameter along the vertical and horizontal axes, respectively.

Simulation

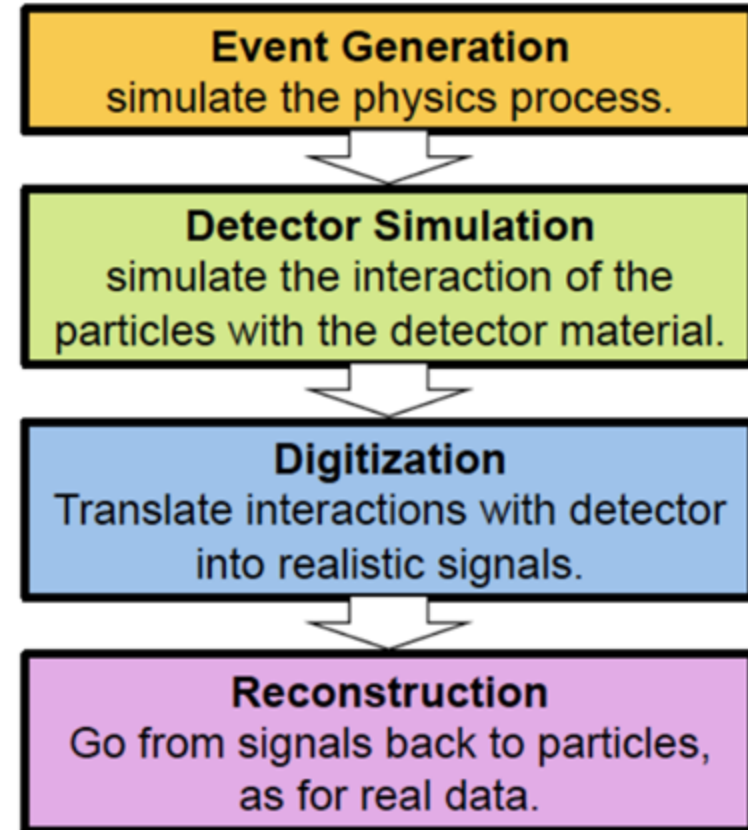
- The core of our studies is comparing hypotheses with the collected data
- For simple systems, we can analytically compute the expected result (given a hypothesis) with the data
- For more complex systems, in which many stages and processes are taking part to the outcome, this is simply not possible
 - Generate artificial events resembling real data as closely as possible
 - Needed for background studies, corrections, error estimation
 - Crucial to guide the design of new detectors / facilities






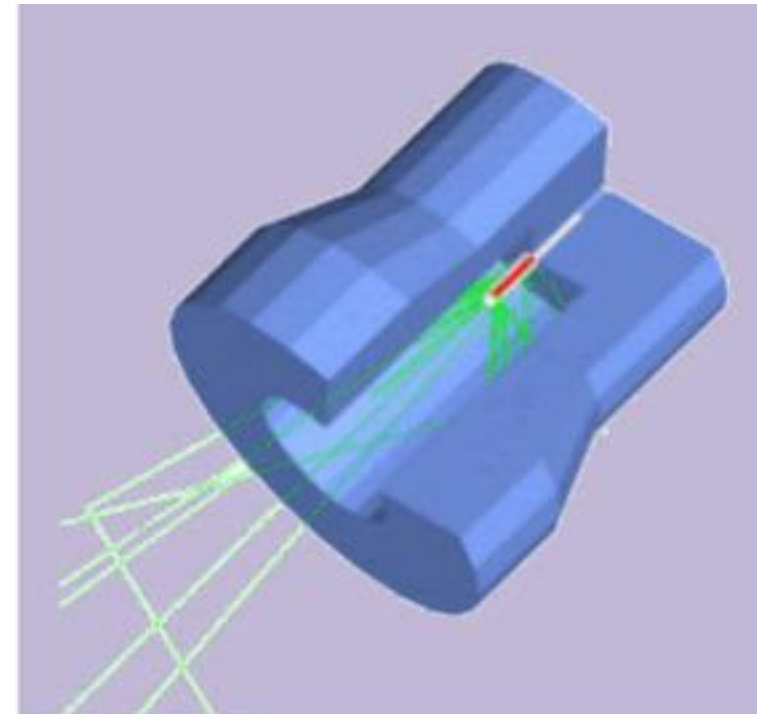
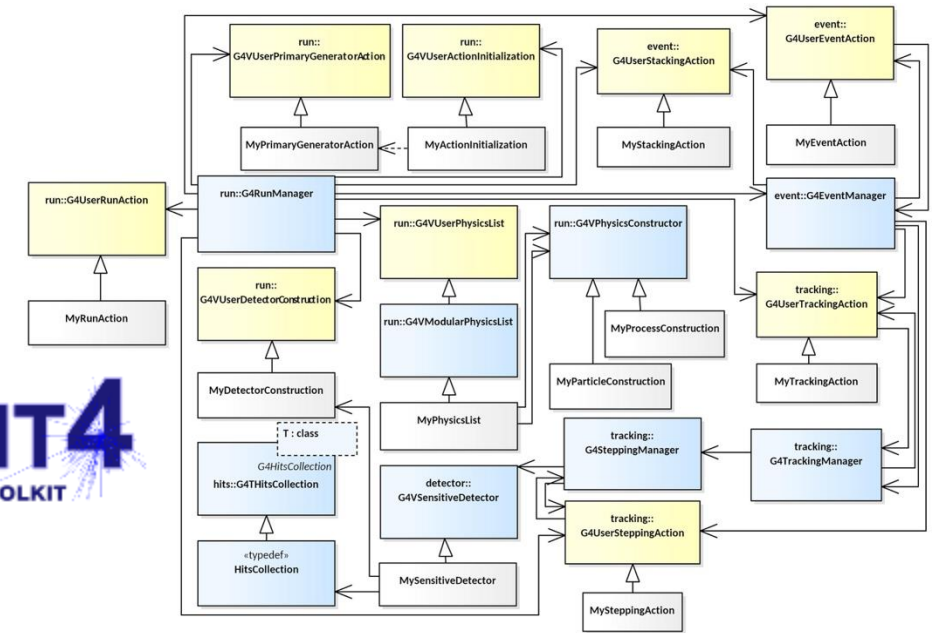
Simulation steps

- **Event generation**
 - Generate particles according to physics of the collision
 - General-purpose and specialized generators
- **Detector simulation**
 - Track particles through the detector, using detector geometry and magnetic field
 - Simulate interaction of particles with matter
 - Generate signals in sensitive volumes
- **Digitization**
 - Simulate digitization process (ADC or TDC)
 - Simulate trigger response
- **Reconstruction**
 - Treat simulated events exactly as real events
 - Keep (some) truth information: association of hits to tracks, association of tracks to vertices, true track parameters, true vertex parameters, ...
 - Store everything



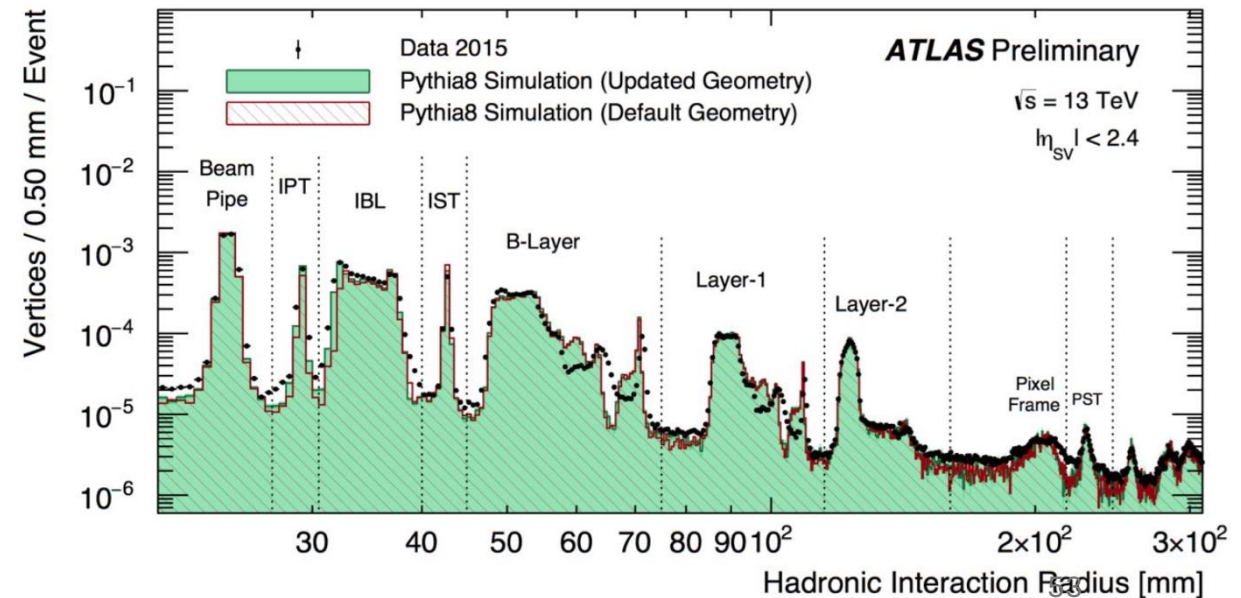
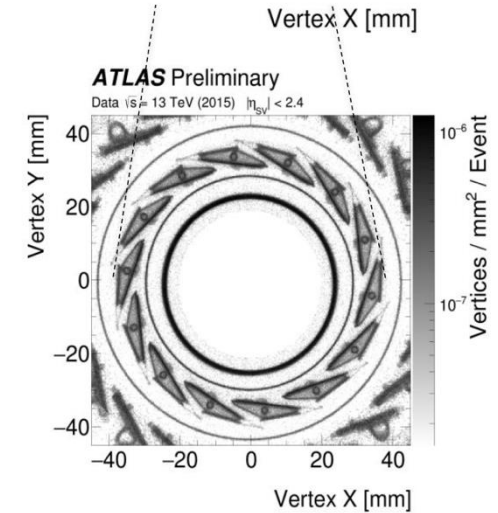
Detector simulation

- **GEANT4: the widely used standard**
 - Object oriented, C++
 - Extremely general and versatile
 - **Implements detailed models of particle interactions with matter in a wide energy range**
 - **Needs detailed description of the apparatus (sensitive and insensitive parts)**
 - **Geometry**
 - Partition the detector into a hierarchy of volumes
 - Describe their shape and their position relative to a mother volume
 - Use possible symmetries
 - **Material**
 - Chemical composition, density
 - Physical properties: radiation length, interaction length, ...
- 



Simulation for detector understanding

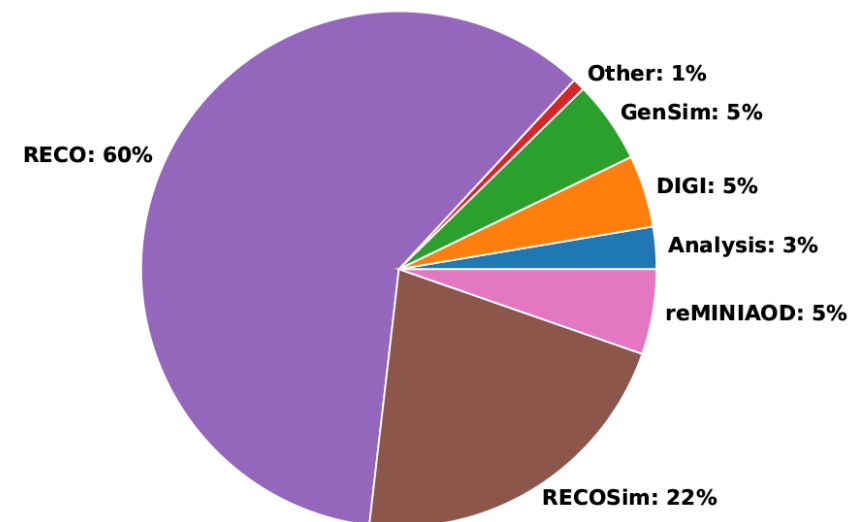
- Use software **simulations** to model the detector as **accurately** and **precisely** as possible based on our best understanding of the physics involved
- **Test** accuracy of simulations using real data
- **Correct** simulations if necessary
- Once simulation gives an **accurate detector model**, it can be used to **correct the data for detector response**



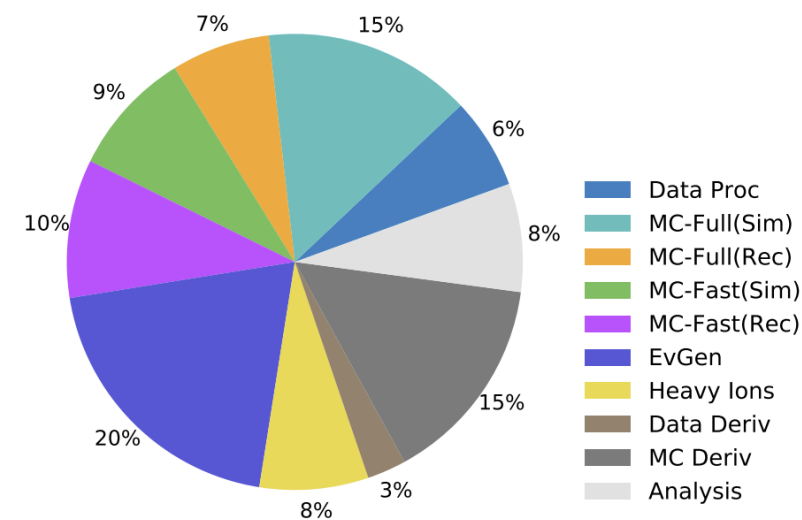
Where do we spend CPU work?

- Different experiments have different shares in the CPU utilization, but in general **simulation** (from partons to electronic signals) and **reconstruction** (from electronic signals to “physics objects” like jets, leptons,) are the most time consuming
- As a rule of thumb, # of simulated events > # of collected events

CMSPublic
Total CPU HL-LHC fractions
2020 estimates

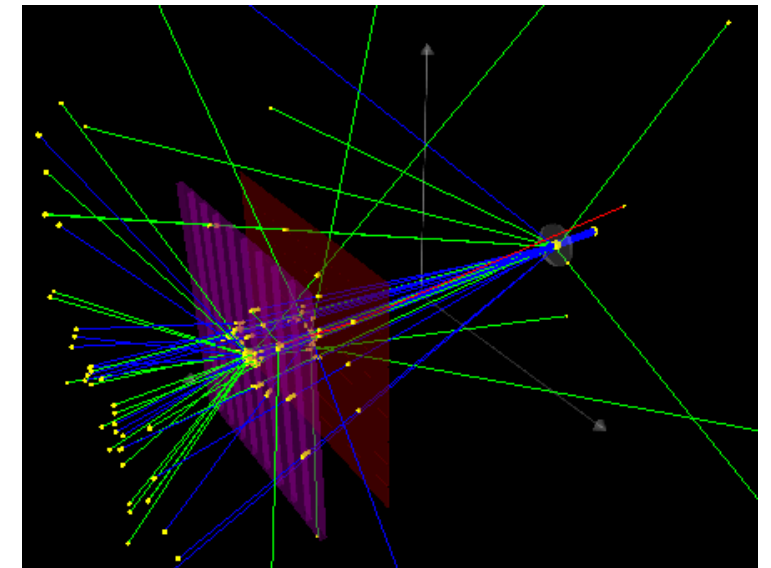
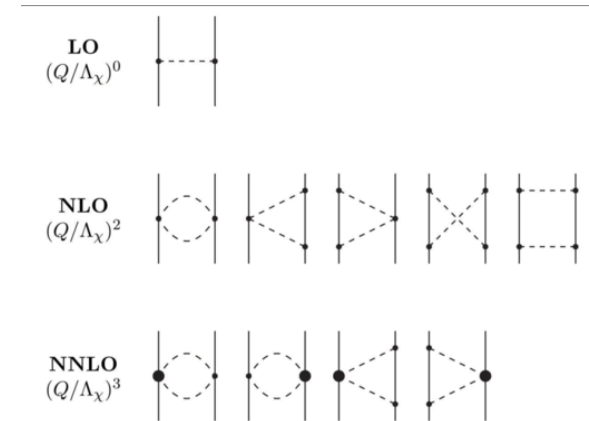


ATLAS Preliminary
2020 Computing Model -CPU: 2030: Baseline



Towards absolute numbers

- **Event Generation:** depends strongly on the generator chooses (Madgraph vs Sherpa vs PowHeg vs ...) and the precision requested (LO vs LNO vs NLO vs ...)
- **Simulation:** by now, the vast majority (all?) the experiments use **Geant4** as the simulation toolkit; still, its requested resources depend on stuff like:
volume of the detector, number of volumes, intrinsic detector resolution, importance of low energy secondary interactions, capability to use parametrization
- **Reconstruction:** The most time consuming task is charged particle tracking using very high resolution detectors (e.g. thin silicon layers). It is a good example since it is *mathematically complex and Highly combinatorial*



But before giving absolute numbers .. unit of measurement for CPU!

- The “**number of CPU seconds**” a task needs is not a proper unit of measurement for CPU, even more if we want to compare results from CPU generations distant in time
- Even industry standard benchmarks (SpectInt, SpecFP, ...) are not suitable, since they probe CPU aspects not necessarily interesting to us
- HEP (via HepiX) created a synthetic benchmark based on a subset of SPEC® CPU2006, which was in use in 2009-2023: HepSpec06 (HS06)
- a new, improved synthetic benchmark based on a weighted average of workflow from HEP experiments was deployed in 2024 and used since then
 - **Rule of thumb:** a CPU “core” today is ~10-20 HS23
 - Hence, a 128 core CPU is ~ 2000 HS06
 - **Hence, a 2 CPU box is today ~ 4000 HS06**

Absolute numbers

- Today, with standard Run-2 LHC, typical numbers are
 - Event generation: 100-1000 HS23.s (which means ~ 10 -100 sev/ev on a single CPU core)
 - Simulation (G4): 500-3000 HS23.s
 - Reconstruction: 150-300 HS23.s
 - Analysis: can be anything, usually quite fast (< 1 -100 HS23.s)
- With these numbers, we can try and project the Computing (CPU and storage) needs for a HEP experiment today, assuming that LHC collides beams ~ 7 Ms/y and 4 experiments

Estimate for a single data taking year

Storage

Data:

7 PB RAW (x2 for a backup copy)
3.5 PB reconstructed data

MonteCarlo

14 PB RAW
7 PB reconstructed simulation

TOTAL ~30 PB/year

CPU

Data:

$7e9 \text{ ev} * 300 \text{ sec} * \text{HS23/ev} = 2e12$
 $\text{sec} * \text{HS23} = 70000 \text{ HS23} * \text{year} (\rightarrow 7000$
CPU cores)

MC

70000 HS23 reconstruction
 $7e9 \text{ ev} * 2500 \text{ sec} * \text{HS23/ev} = 1.7e13 = 500000$
HS23*year simulation

Analysis (MC + DT):

$7e9 \text{ ev} * 2 * 10 \text{ sec} * \text{HS23/sec} * N = 1.4e11$
 $\text{sec} * \text{HS23} * N = 4500 * N \text{ HS23}$
Where N is the number of independent
analyses, can be very high (~100)

TOTAL ~ 1.1M HS23

...corresponding to 3000 HDD/y 100000 computing cores per experiment!

In reality

| | CPU [kHS23] | Disk [PB] | Tape [PB] |
|-------|-------------|-----------|-----------|
| Tier0 | 2690 | 201 | 825 |
| Tier1 | 3648 | 414 | 1072 |
| Tier2 | 4421 | 431 | |
| Total | 10759 | 1045 | 1897 |

- The estimate in the last page does not account for the fact that multiple years are used at the same time, mistakes are done, special data taking periods also take resources. And, on top of that, there are always (at least) 3 activities going on
 - Analyzing data from previous + current year
 - Taking data in the current year
 - Preparing future data taking periods and detector upgrades
- So, all in all, real resource number per experiment are **underestimated by at least a factor 3x**

How to handle this?

- By today's metric, handling some 1 Million CPU cores and 2-3 Exabytes of data does not seem an impossible task
- But, LHC was approved in the mid 90s, when 1 single HDD was 10 GB, and a CPU was probably 0.1 HS06
- You can understand what **leap of faith in technology** is needed to think that in 10 years (the expected start of LHC was < 2005) you will be able to handle resources which, in 1995, were of the same size of the entire world IT resource



How to design a computing model for HEP in ~ 1995?

- **Build a BIG data center**

- A large building with ~1000000 computing cores, and 200000 HDD; Probably it would work; **Google** apparently has facilities much larger than that; **NSA** for sure...
- But: It would be a single point of failure; problem finding enough personnel in a single area, member states not willing to fund resources abroad, ...

- **Use many small data centers**

- De-localized cost / expertise / redundancy; member states happy since they can build a local infrastructure, ...

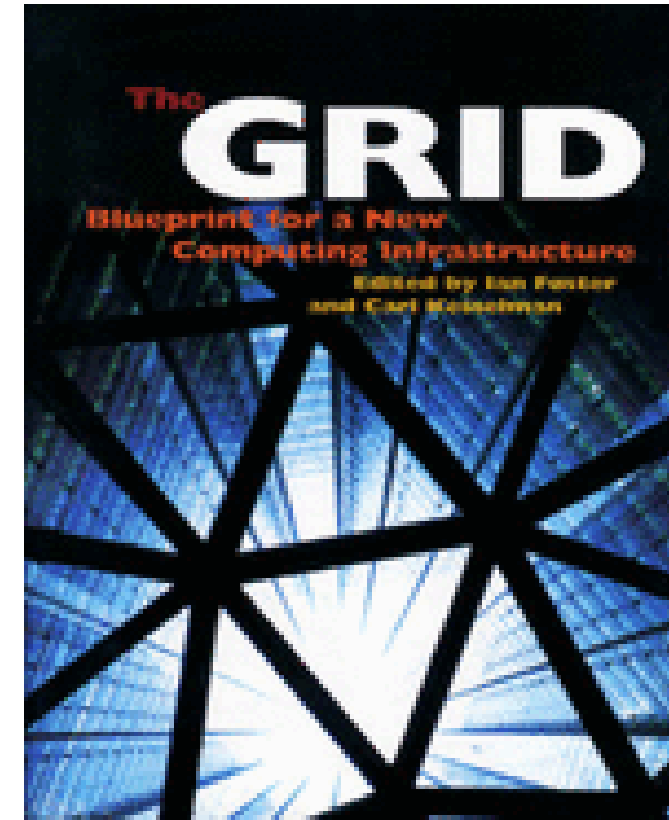
Introducing the GRID

- Idea was not new in Computer Science; HEP had “simply” to make it real at a large scale

geographical

“When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special purpose appliances”

(George Gilder)



The idea in a nutshell

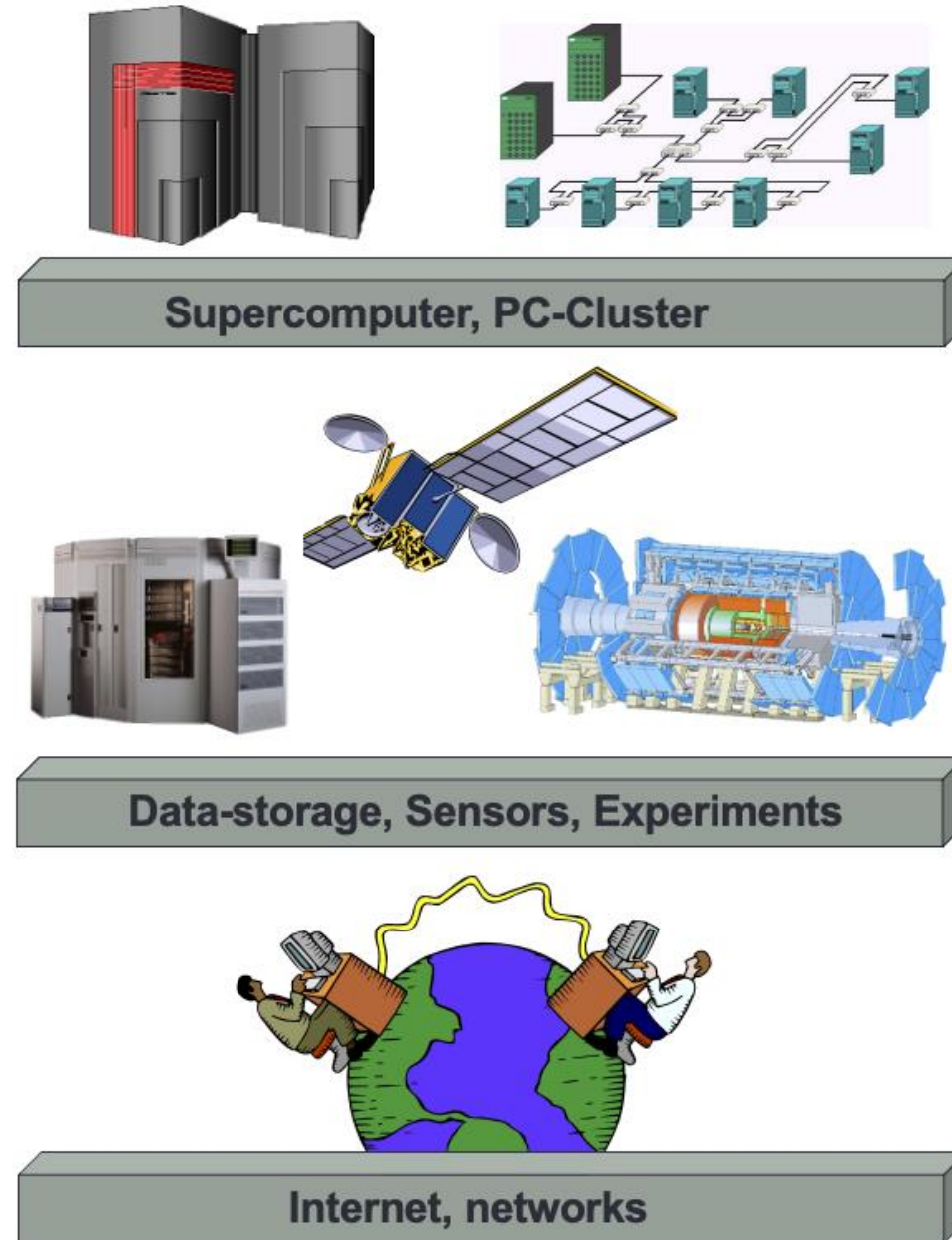
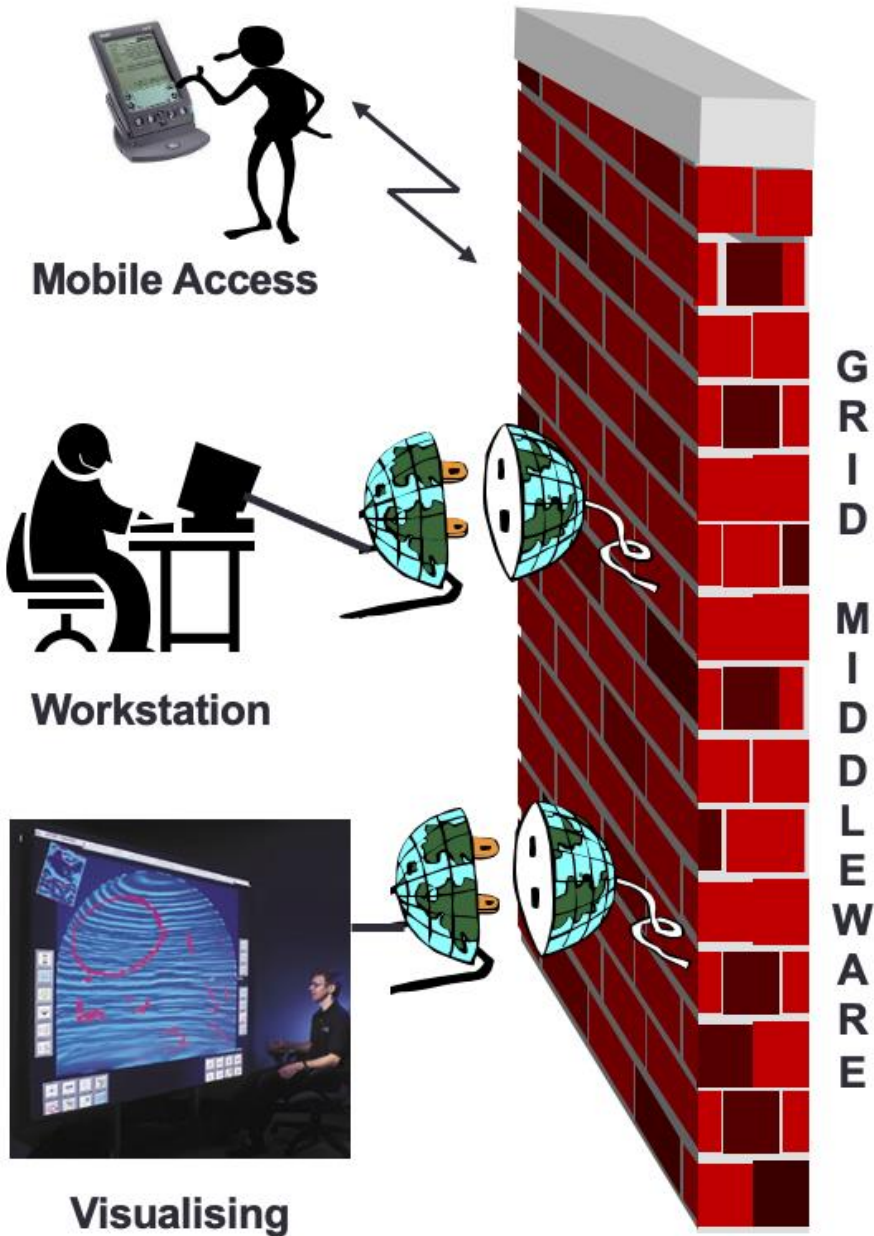
Split the problem into two levels:

- **The physical level:**

- Distribute resources worldwide in N (>100) centers
- Technically a nightmare: distributed Authentication, Authorization, network paths, multiple access protocols to CPUs/Storage, ...

- **The logical level:**

- Try and provide the users (the physicists!) with a logical single view, where "many CPUs" and "a lot of storage" is available in a "flat view"



Build a wall
(call it API
layer,
intelligent
system, ...)

The implementation

Leaving aside the historical development, we have now

- A global entity for LHC computing (and more, see later), the Worldwide LHC Computing GRID (WLCG) – sometimes called the “5th big LHC Collaboration”
- A set of low-level tools allowing the collaboration to work:
 - A trust model for mutual Authentication and Authorization
 - A set of recognized protocols for data access, data movement, metadata organization, support, accounting
- O(200) centers in the collaboration
 - With “guaranteed” service levels and some obligations...



WLCG
Worldwide LHC Computing Grid

Accounting Portal



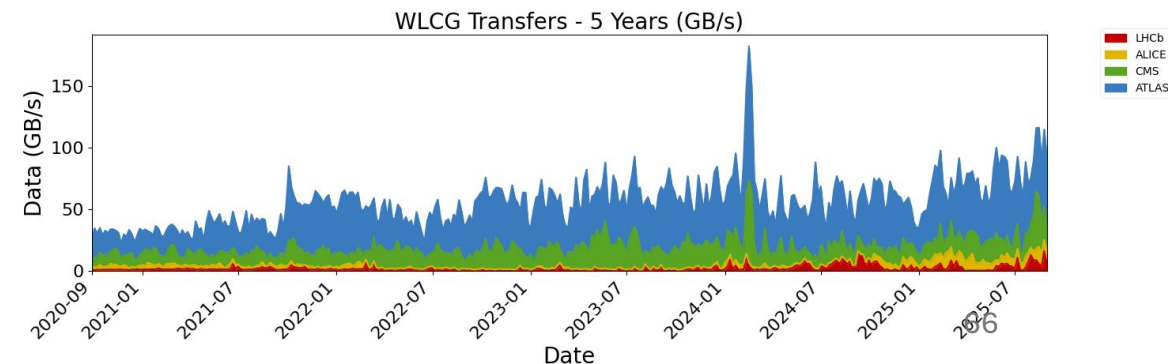
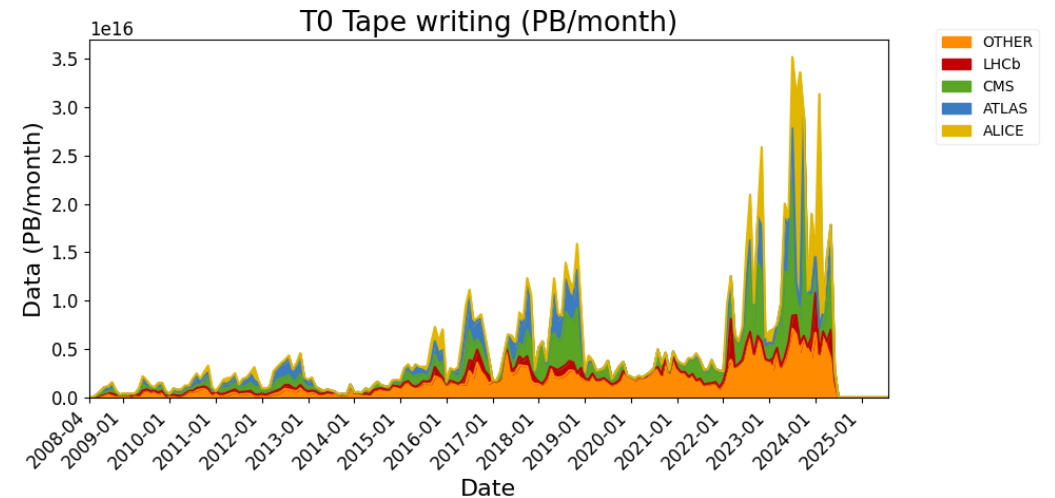
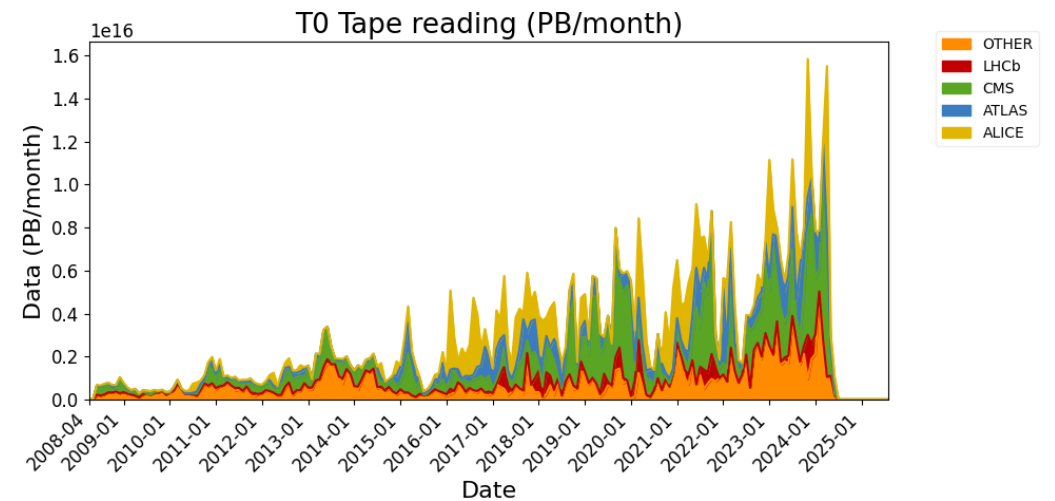
X.509

iam



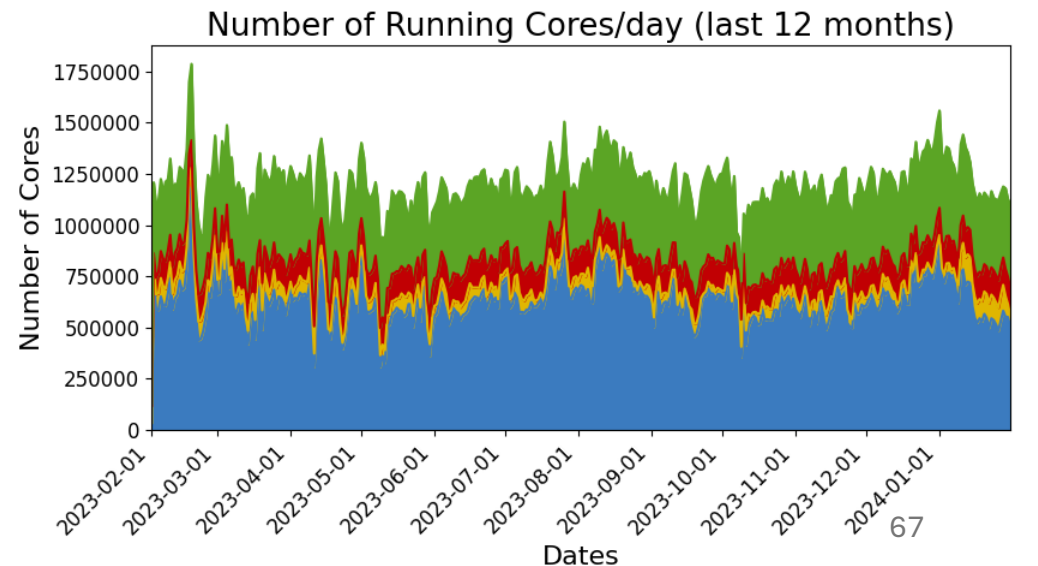
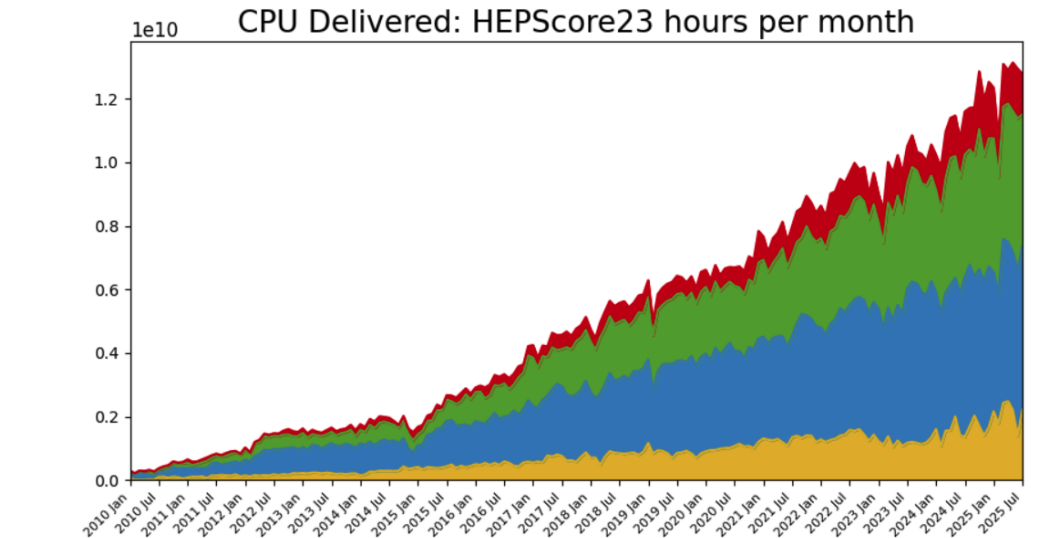
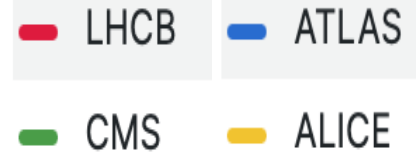
Data management

- Dealing with exabytes of data and a complex distributed computing infrastructure. Data are stored
 - on disk for immediate access
 - on tape for archive
 - access takes longer but it is much cheaper, so we can store much more data
- Data management systems catalog data and track the location of data (site A, B, C ...)
 - Rucio and DIRAC are prominent examples of data management systems
- Data management systems can also initiate transfers between sites



Workflow management

- Multi-dimensional optimization problem : orchestrating work accessing data and producing derived data
- Submission infrastructure software (like HTCondor) and workflow management software (Like Panda, Dirac, WMAgent) automate
 - Job creation
 - Job execution
 - Job monitoring and failure recovery

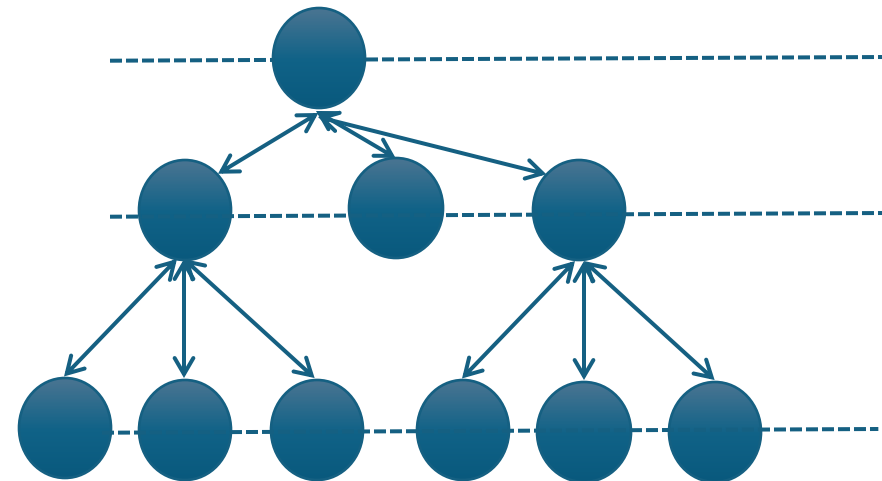
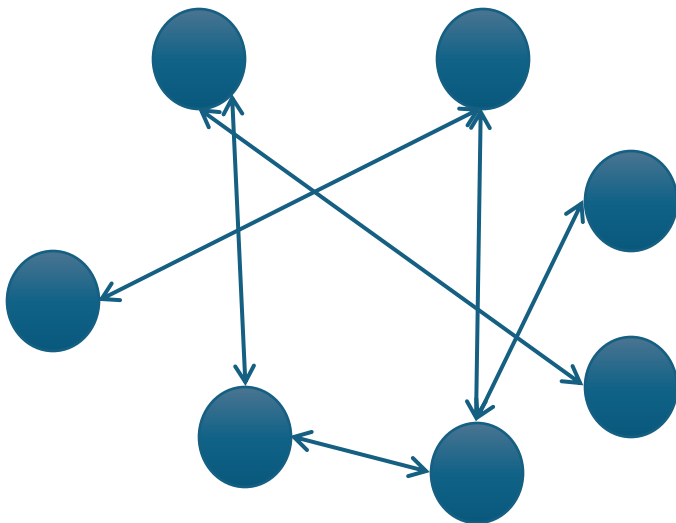


Authentication & Authorisation

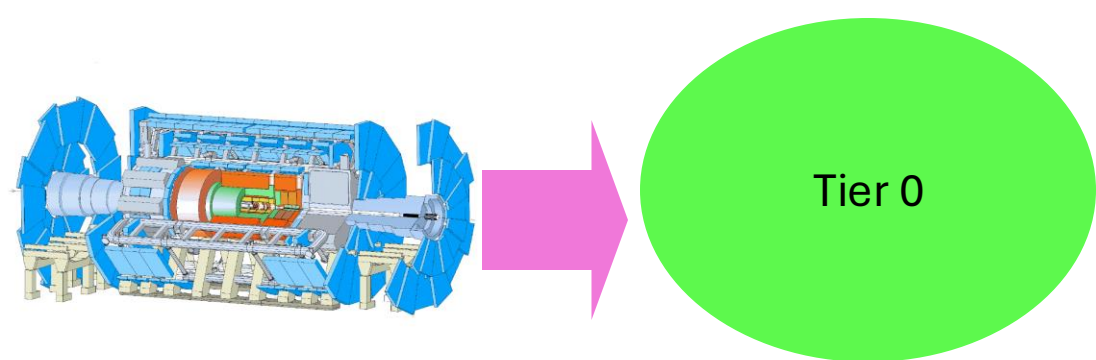
- Mutual trust and AAI is the most important building block:
 - As a LHC scientist, you can literally access resources in every corner of Earth
 - It is the cornerstone on which the various access protocols will be based upon. We started with X509 certificates, we are transitioning to Indigo-IAM tokens
- If you have a distributed infrastructure which you want to use as a big single entity, the technical building block you need is “tons of network” (aka have a *WAN as fast as LAN*)

The network

- The ideal “as if local” is possible when all the nodes see all the data at “as local” speed; which in LHC metrics mean ~ **each** core should be able to access **every** piece of data at $O(5 \text{ MB/s})$
- In 1995 this was a dream: network lines are expensive and rare (*no Netflix yet!*); we cannot assume to prepare the full mesh of networking for $O(100)$ centers – which would mean **$n(n-1)/2$ connections $\rightarrow O(10^4)$**
- **MONARC** project studied and proposed a hierarchy of computing centers: the “**Tiered data model**”; fewer paths are needed, and their importance is different



tiers

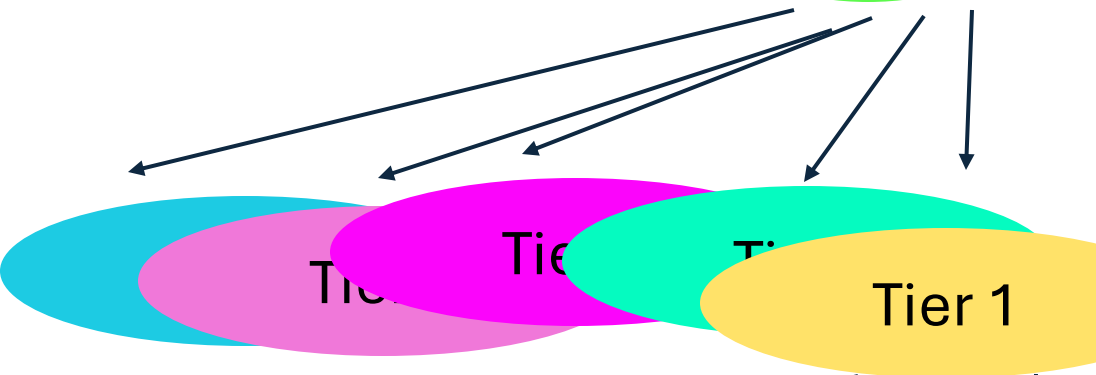


CERN

Master copy of RAW data

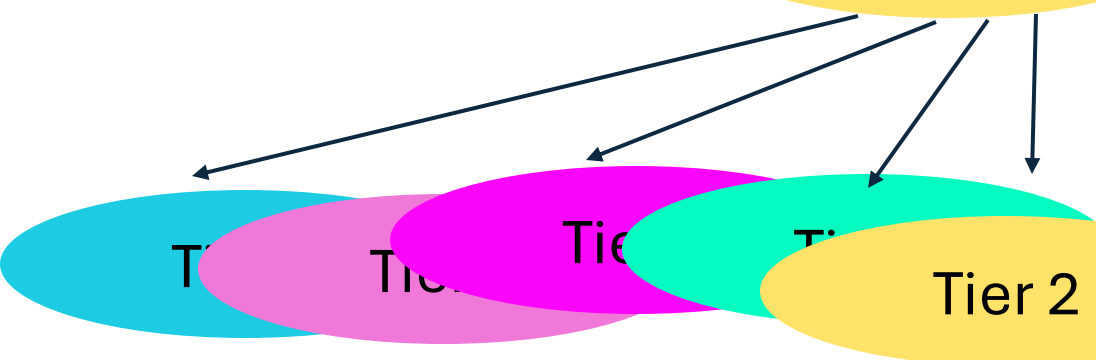
Fast calibrations

Prompt Reconstruction



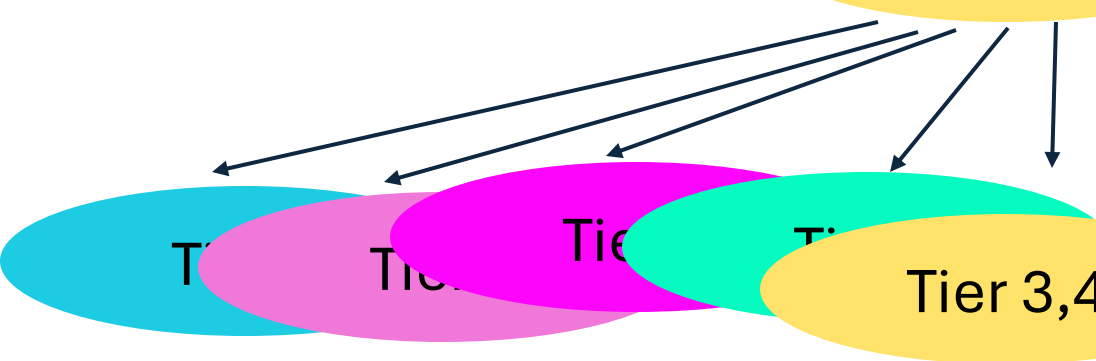
A second copy of RAW data (Backup)

Re-reconstructions with better calibrations



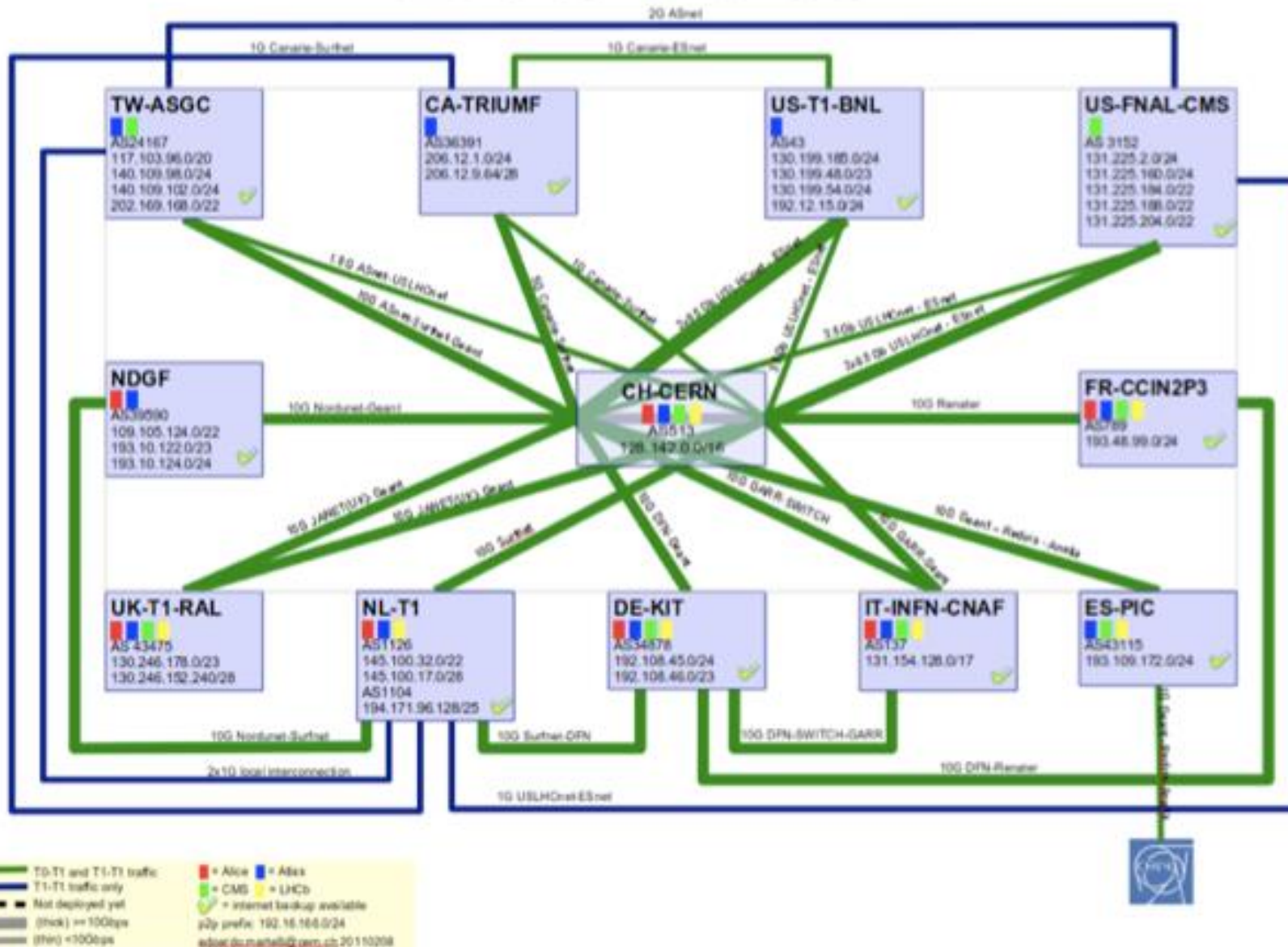
Analysis Activity

They are dimensioned to help ~ 50 physicists in their analysis activities



Anything smaller, from University clusters to your laptop

LHCOPN



1st need: **put the data in safety**


1st copy stays @ CERN, but a 2nd copy must go distributed for disaster recovery

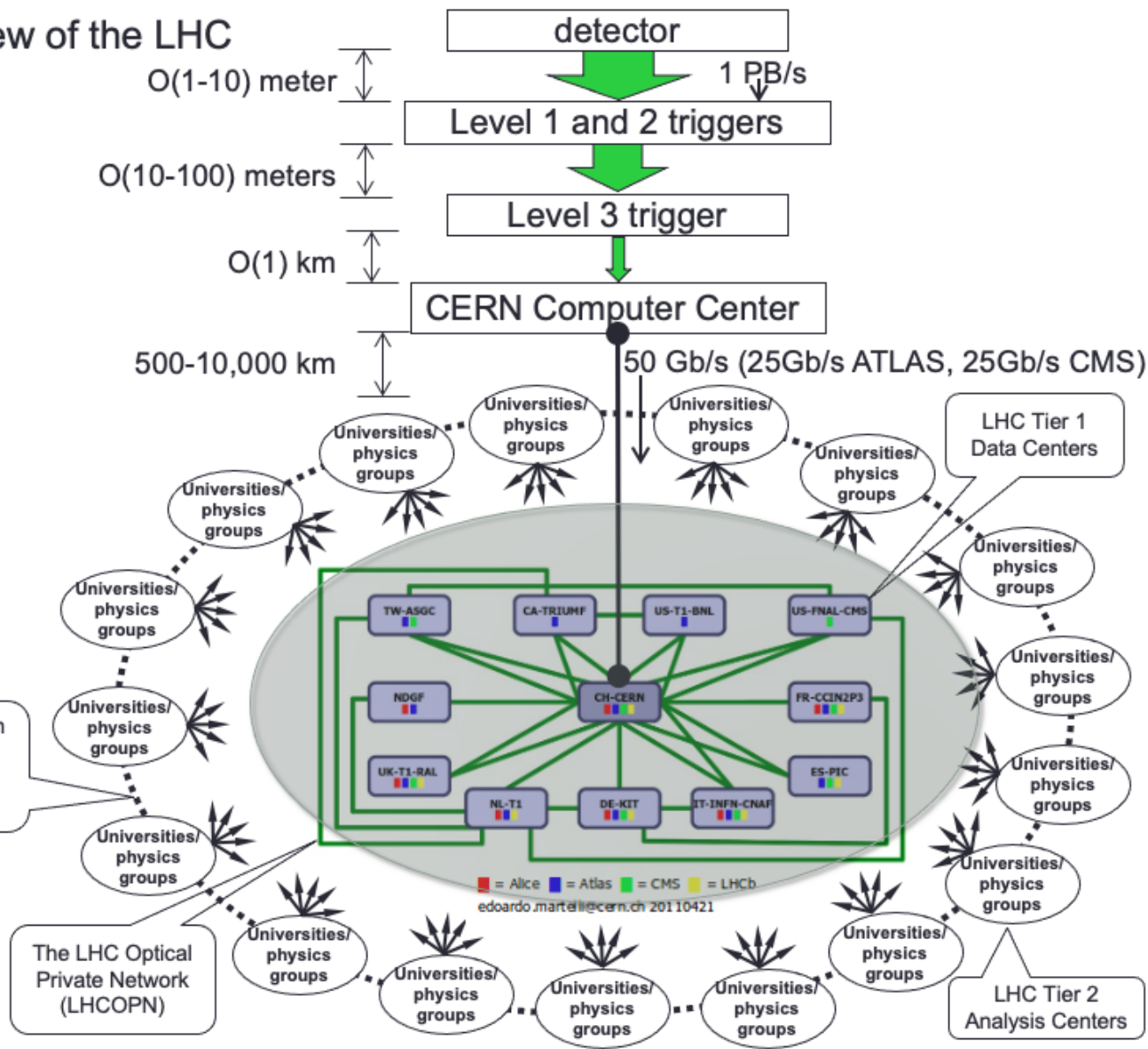
→ **Guaranteed lines Tier-0 → Tier-1s**

→ **By today , multiple of 100 Gbps**

A Network Centric View of the LHC

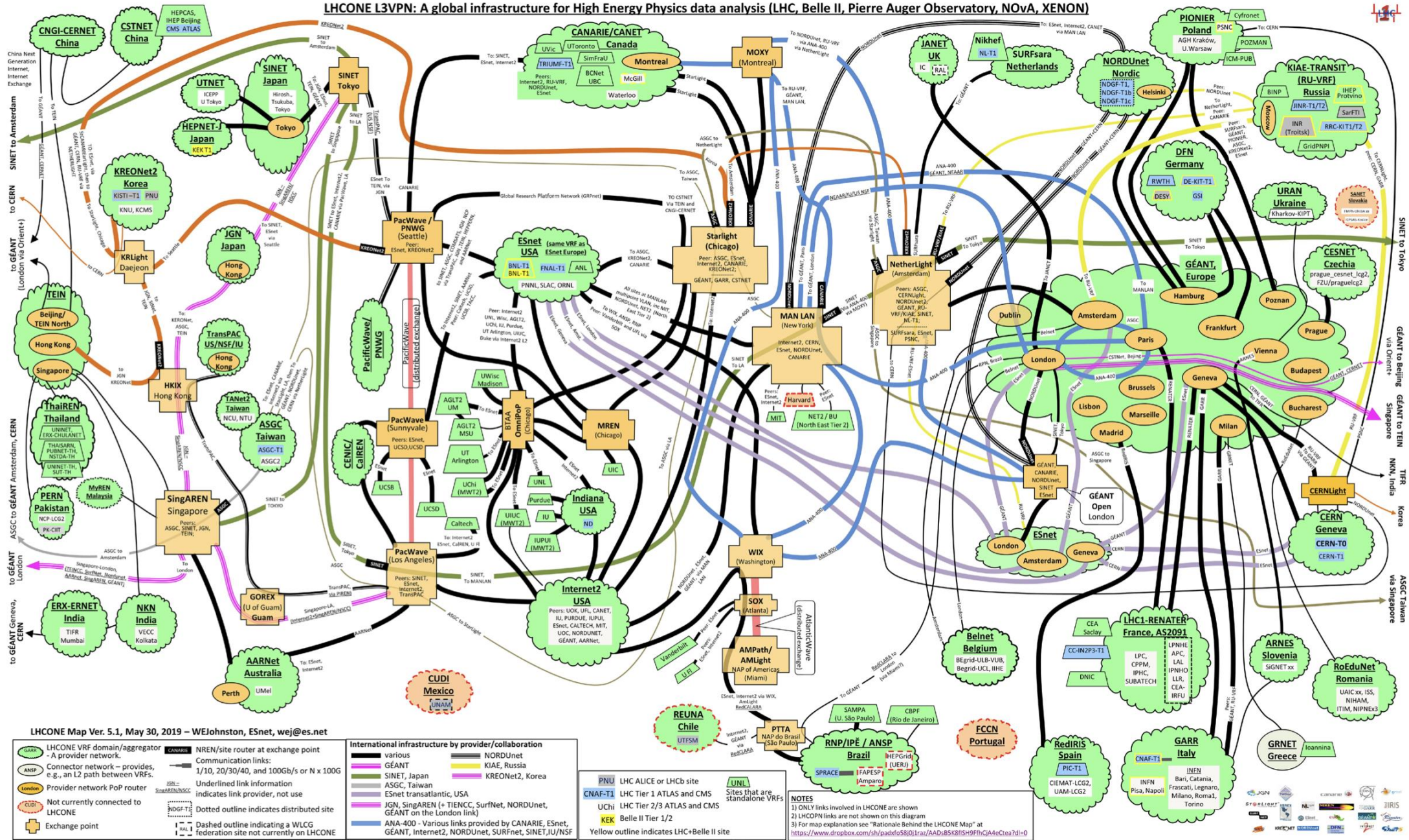
| CERN →T1 | miles | kms |
|----------------|-------|------|
| France | 350 | 565 |
| Italy | 570 | 920 |
| UK | 625 | 1000 |
| Netherlands | 625 | 1000 |
| Germany | 700 | 1185 |
| Spain | 850 | 1400 |
| Nordic | 1300 | 2100 |
| USA – New York | 3900 | 6300 |
| USA - Chicago | 4400 | 7100 |
| Canada – BC | 5200 | 8400 |
| Taiwan | 6100 | 9850 |

This  is intended to indicate that the physics groups now get their data wherever it is most readily available

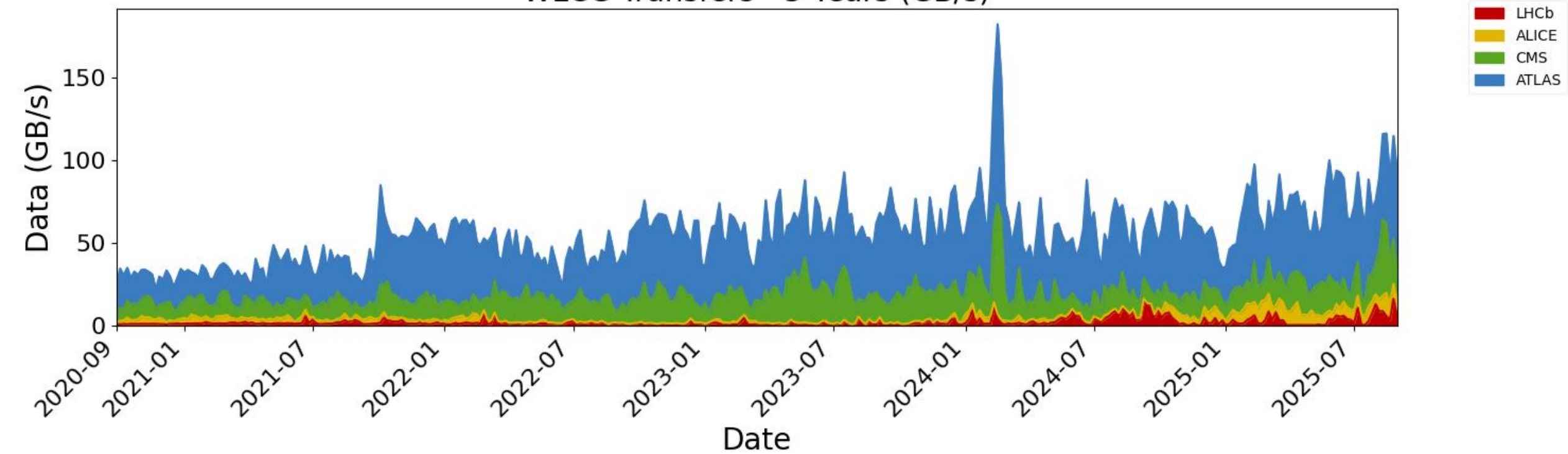


... and from Tier-1s to the other data centers ...

LHCONE L3VPN: A global infrastructure for High Energy Physics data analysis (LHC, Belle II, Pierre Auger Observatory, NOvA, XENON)



WLCG Transfers - 5 Years (GB/s)



Tiered model: the distributed system must be smarter!

- Optimize transfers, avoid too many “jumps”
- Moving data is expensive / time consuming → **move** the jobs to data, and not vice versa
- ... which means at some point you need to have the “most important data” in the “best places” → need for **smart** data placement, data lifecycle, multiple copies, caches, ...

Software

HEP collaborations have quite unique needs for software:

- It is inevitably large → see later
- It must be runnable on every country participating the effort, and more → no copyrights, no embargoed code
- It must cover a large range of use cases → simulation, reconstruction, selection, analysis, ...
- It is a long journey; experiments last $O(10-30y)$, difficult to rewrite from scratch when taking data

Software stack



Main languages used:

- **C++** for the physics data processing framework and some analysis software
- **Python** for the analysis and scripting/configuration

But other languages too: C/C++ used for interfacing with electronics, VHDL for FPGA, Julia, etc...

More than 5 million lines of code per experiment, development started in the early 2000s. Covers the code to integrate with the DAQ, data analysis, event reconstruction,....

Using **frameworks** to process the events (Athena/Gaudi, CMSSW, ROOT)

Using many **external tools and libraries** when needed: e.g. BOOST for C++, BLAS, Eigen for linear algebra, Tensorflow, Catboost for ML...



How big?

- **SLOC** are a standard industry metric, and there are tools to translate them into «man years» and in the end to \$\$ (assuming a US typical programmer)
- The result is enormous, but reflects the fact that both software stacks are 15 years old or more
- It does NOT include externals, like Geant4, geometry engines, particle generators, ROOT, etc

Table 6. SLOCCount measured lines of source code for ATLAS and CMS.

| Experiment Type | Source Lines of code (SLOC) | Development effort (person-years) | Total estimated cost to develop |
|--------------------|--------------------------------|--------------------------------------|------------------------------------|
| ATLAS | 5.5M | 1630 | 220 M\$ |
| CMS | 4.8M | 1490 | 200 M\$ |

- As a reference:
 - **Linux Kernel** is: 15M sloc, 4800 FTEy, 650M\$ (3x CMS)
 - **Geant4** is: 1.2M sloc, 330 FTEy, 45 M\$ (1/4x CMS)

.. But this is only the "core code"

- We rely on many externals (Geant4 is an external, ROOT is an external, Pythia is an external) which inflate greatly the total size
- This (in unreadable fonts) is the list of externals for a typical CMS release

alpgen qd root_cxxdefaults sockets catch2 gcc-cxxcompiler gcc-cxxcompiler gcc-f77compiler mpfr cmsswdata codechecker csctrackfinderemulati cuda-stubs cuda-gcc-support cvs2git dablooms db6 dmtcp doxygen eigen fastjet-contrib fastjet-contrib-archi gcc-analyzer-ccompile gcc-analyzer-cxxcompil gcc-atomic gcc-checker-plugin gcc-plugin gdb geant4-parfulicms geant4data py2-numpy openloops git glibc glimpse gmake gnuplot gosam gosamcontrib hdf5 igprof intel-license itnotify lapack lcov libffi libxslt llvm md5 openblas ofast-flag openmpi professor py2-sympy py2-absl-py py2-appdirs py2-argparse py2-asn1crypto py2-atomicwrites py2-attrs py2-autopep8 py2-avro py2-awkward py2-backcall py2-backports py2-backports-functools py2-backports-abc py2-beautifulsoup4 py2-bleach py2-bokeh py2-bottleneck py2-cachetools py2-certifi py2-cffi py2-chardet py2-click py2-climate py2-colorama py2-contextlib2 py2-cryptography py2-cx-oracle py2-cycler py2-cython py2-dablooms py2-decorator py2-defusedxml py2-docopt py2-downhill py2-dxr py2-entrypoints py2-enum34 py2-flake8 py2-flawfinder py2-fs py2-funcsigs py2-functools32 py2-future py2-futures py2-gast py2-gi-tdb2 py2-gi-tpython py2-google-common py2-google-packages py2-grpcio py2-h5py py2-h5py-cache py2-hep_ml py2-histbook py2-histogram py2-html5lib py2-hyperas py2-hyperopt py2-idna py2-ipaddress py2-ipykernel py2-ipython py2-ipython-genutils py2-ipywidgets py2-jedi py2-jinja2 py2-js-onpickle py2-jsonschema py2-jupyter py2-jupyter-client py2-jupyter-console py2-jupyter-core py2-keras py2-keras-application py2-keras-preprocess py2-kiwisolver py2-lint py2-lizard py2-llvmlite py2-lxml py2-lz4 py2-markdown py2-markupsafe py2-matplotlib py2-mccabe py2-mistune py2-mock py2-more-itertools py2-mpld3 py2-mpmath py2-nbconvert py2-nbdime py2-nbformat py2-networkx py2-neurdlab py2-nose py2-nose-parameterize py2-notebook py2-numba py2-numexpr py2-oamap py2-onnx py2-ordereddict py2-packaging py2-pandas py2-pandocfilters py2-parsimonious py2-parsio py2-pathlib2 py2-pbr py2-pexpect py2-pickleshare py2-pillow py2-pip py2-pkgconfig py2-plac py2-pluggy py2-ply py2-prettytable py2-prometheus-client py2-prompt-toolkit py2-protobuf py2-prwlock py2-psutil py2-ptyprocess py2-py py2-pyasn1 py2-pyasn1-modules py2-pybind11 py2-pybrain py2-pycodestyle py2-pycparser py2-pycurl py2-pydot py2-pyflakes py2-pygit2 py2-pygments py2-pymongo py2-pyopenssl py2-pyparsing py2-pysqllite py2-pytest py2-python-cjson py2-python-dateutil py2-python-ldap py2-pytz py2-pyyaml py2-pyzmq py2-qconsole py2-rep py2-repoze-lru py2-requests py2-root_numpy py2-root_pandas py2-rootpy2-scandir py2-schema py2-scikit-learn py2-scipy py2-seaborn py2-send2trash py2-setuptools py2-simplegeneric py2-singledispatch py2-six py2-smmap2 py2-soupsieve py2-sqlalchemy py2-stevedore py2-subprocess32 py2-tables py2-tensorflow py2-terminado py2-testpath py2-theano py2-theano py2-thrift py2-tornado py2-tqdm py2-traitls py2-typing py2-typing_extensions py2-uncertainties py2-uproot py2-uproot-methods py2-urllib3 py2-virtualenv py2-virtualenv-clone py2-wcwidth py2-webencodings py2-werkzeug py2-wheel py2-widgets nbextensio py2-xgboost py2-xrootdpyfs pydata pyminuit2 pyqt python-paths python_tools rootglew scons sloccount tcmalloc tcmalloc_minimal tensorpy2-virtualenvwrapper flow tinyxml2 xtl blackhat boost boost_header python bz2lib cascade_headers ccache-cxxcompiler ccache-cxxcompiler ccache-f77compiler zlib gmp photos_headers pythia6_headers openssl clhep clhepheader cppunit cuda curl libxml2 dcap root_interface xz xerces-c vecgeom_interface hepmpc_headers distcc-cxxcompiler distcc-cxxcompiler distcc-f77compiler dpm expat fastjet fftjet fftw3 freetype gbl gdbm gsl glib google-benchmark libjpeg-turbo hector heppdt madgraph5amcatnlo llvm-cxxcompiler jemalloc jimmy_headers ktjet libhepml libuuid llvm-cxxcompiler llvm-f77compiler meschach mxnet-predict numpy-c-api x11 oracle pacparser yoda protobuf python3_qd_f_main sqlite sigcpp taula_headers tbb tensorflow-framework tensorflow-runtime tensorflow-xla_compiler0-pafccj3 toprex_headers utm valgrind vdt_headers xrootd.tensorboost_system boost_boost_iostreams boost_serialization boost_program_options boost_python boost_regex boost_signals boost_test cascade yaml-cpp photos pythia6 pcre cub cuda-api-wrappers cuda-cublas cuda-cufft cuda-curand cuda-cusolver cuda-cuspars cuda-npp cuda-nvgraph cuda-nvjpeg cuda-nvml cuda-nvrtc das_client vecgeom hepmpc frontier_client google-benchmark-main libpng iwyu-cxxcompiler libffi libgfortran llvm-analyzer-cxxcompil llvm-analyzer-cxxcomp modb opengl opendap oradoccl pyclang qtbase sip starlight taula tensorflow-c tensorflow-cc tkonlinesw toprex vdt boost_chrono boost_filesystem boost_mpi cgal lhapdf classlib davix rootcling geant4core photospp geant4static graphviz lwtm millpede qt3support rivet tkonlineswdb cgalimagoio herwig rootmathcore rootpythia8 geant4vis thepeg pyquen qt rootrint rootrlx rootmatrix rootx11 sherpa charybdis rootthread dire taulapp geant4 geneva herwigpp jimmy qt designer rootgeom rootxmlio vincia rootcore evtgen rootthstmatrix rootmath rootxml rootphysics rootpad rootfoam rootspectrum root rootminuit rootgraphics rootgui rootinteractive roothtml rootminuit2 dd4hep-core rootfcore mctester professor2 rooteg rootgeompainter rootgl rootged rootguihtml rootmlp rootpy dd4hep dd4hep-geant4 rootf rooteve rootmva rootstats rootpymva histfactory coral

- Note that **gcc** is there! CMS ships its own compiler, so dependency on the host Linux is only at the level of glibc

The HEP framework(s)

- Such a complexity of use cases and code, with multiple alternatives in each of them, needs a coherent Framework, which is at the core of the HEP software, and is the piece which basically stays stable-with-adiabatic-changes within the experiment lifetime. Changing a FW is not easy; it is not done during data taking.
- Typical needs from a framework
 - **Modularity:** large utilization of plugins, algorithms, external libraries
 - **Scheduling:** must be efficiently able to schedule the execution of code (taking into account dependencies) on the available resources
 - **Portability:** not attached to a single compiler / OS / architecture
 - **Evolution:** the computing scenario is not static. From 2008 to now for many things happened; still most of the FW interface has been stable:

From GRID to Clouds to Virtualization to HPC to heterogeneous computing (GPU, FPGA, QC even...)

From data locality to streaming storage federations

From SL4/gcc4 to CC7/gcc8

From 32 to 64 bit

From single process to multi process to multi threaded

From single core PCs to O(300) cores per PC

From configs to Python as the uber language

From fully scheduled execution to unscheduled (needed for multi threading)

Analysis support from ROOT(cint)-ROOT(cling)-PyROOT-UpROOT

Software Engineering

Requirements on software are strict:

- **Reproducibility of the results** (we need to re-run old versions of the software)
- Tracking the **provenance of the data** is crucial !

The software engineering process is critical to keep the software running, when ~100s of developers potentially modify the code

- Adequate **unit and integration tests** are necessary
- Use of **version control software** (Git)
- **Continuous integration** (Gitlab CI, Jenkins...)

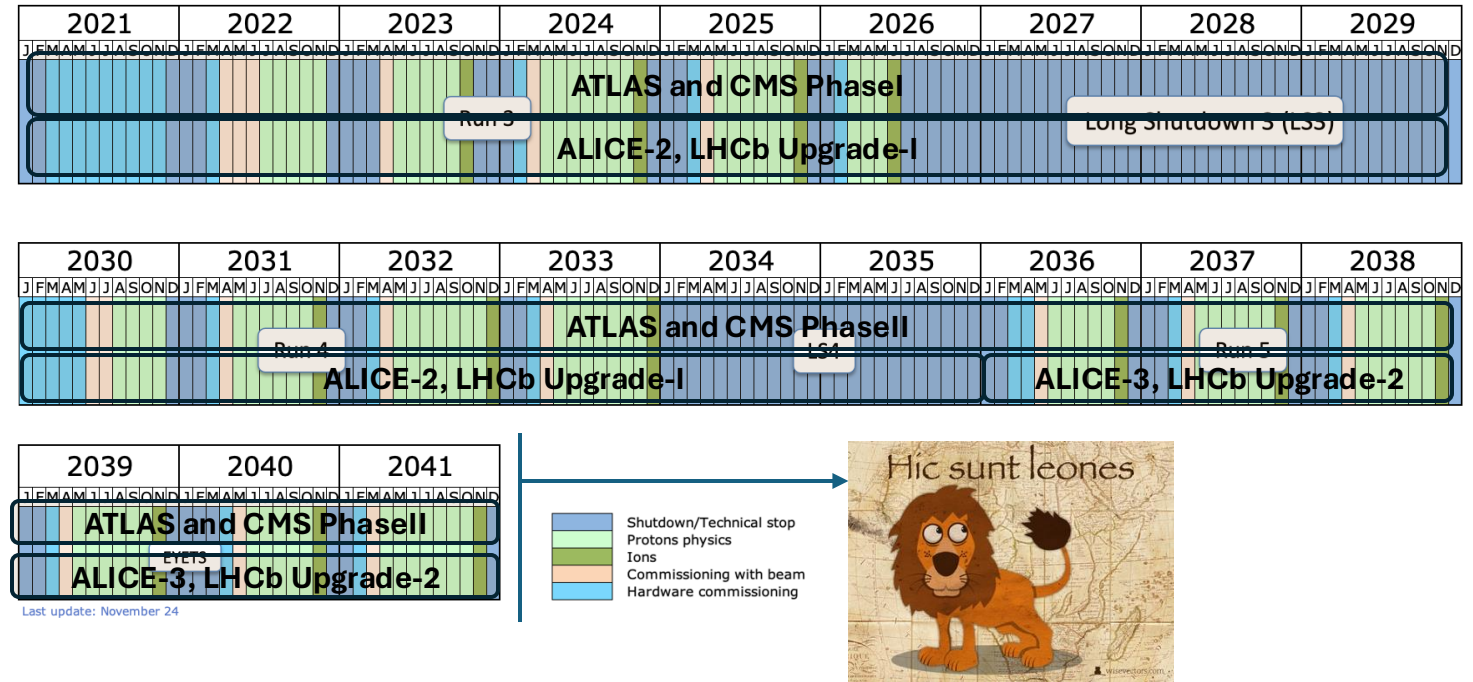


The future

- **“it all works”, so why change?**
- We have the proof that the computing systems for today’s collider experiment do work. The LHC collaborations have published thousands of papers each
- Computing is a large operational cost; but is \sim constant year over year and somehow possible to cover
- **Are we done? No we are not ...**

The expected future within ~a decade

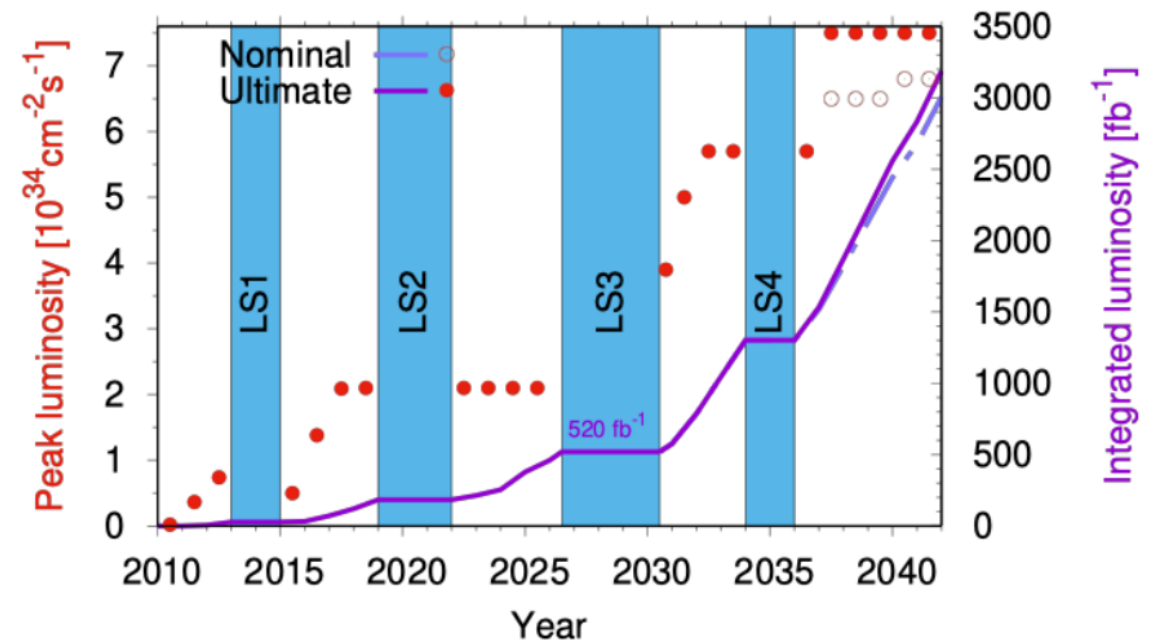
- CMS and ATLAS undergoing the largest upgrades with Run4: 3000/fb total lumi
 - Computing planning and modelling well advanced
- ALICE and LHCb have successfully upgraded for Run3 – gearing up to have upgrades approved for Run5
 - Computing under (early) studies
- Future projects (FCC, ILC, etc) not considered
 - Expect further steps in resource requirements



ATLAS and CMS: nominal inst lumi from $2e34$ to up to $7.5e34$; trigger rates in the $O(10)$ kHz ballpark
LHCb: from $2e33$ up to $1.5e34$
ALICE: 3x Pb-Pb rates, 90x pp

Projections for HL_LHC (ATLAS+CMS)

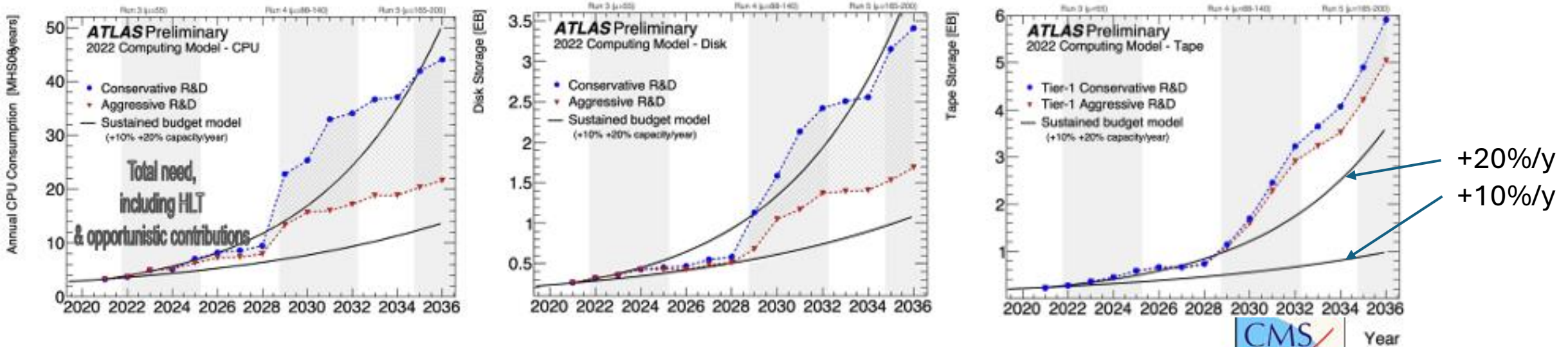
- In the end, main parameters are
 - **Trigger rate**: from 1 to 7.5kHz
 - **Mean number of collisions per bunch crossing (pileup) $\langle \text{PU} \rangle$** : from 35 to 200
 - More and more crowded events
 - increased bandwidth to storage (x42)
 - Impacts storage (\sim linearly) and CPU (superlinearly)
 - **Live time of the Accelerator**
 - **Monte Carlo production needs**
- Expect naïve scaling of x50-x100



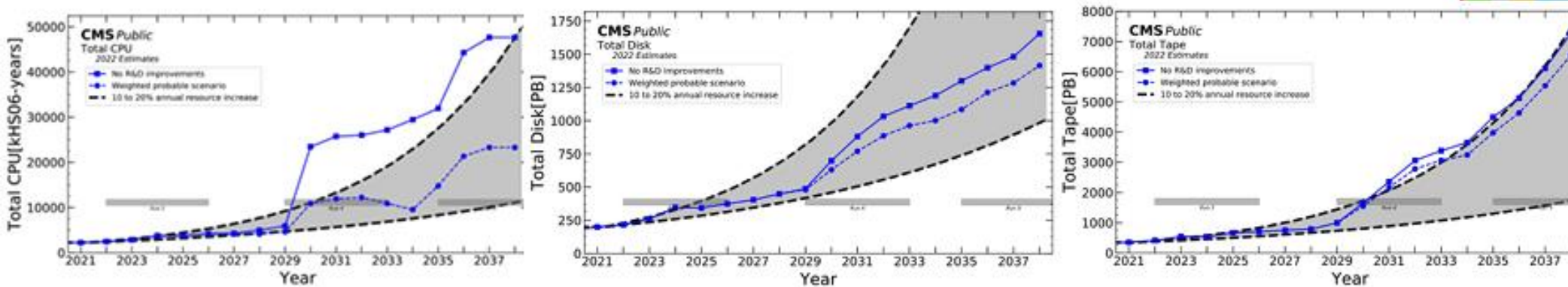
Some true but amazing statements:

- “We collected 5% of LHC foreseen integrated luminosity”
- “We are at 1/5th of the LHC machine capabilities”

ATLAS and CMS [latest public projections]

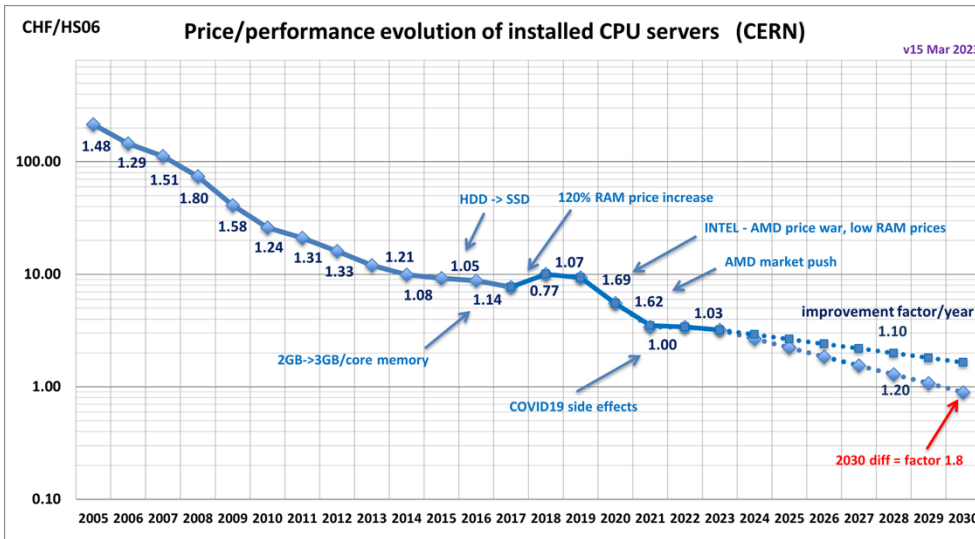


CMS offline resource needs for Run4/Run5



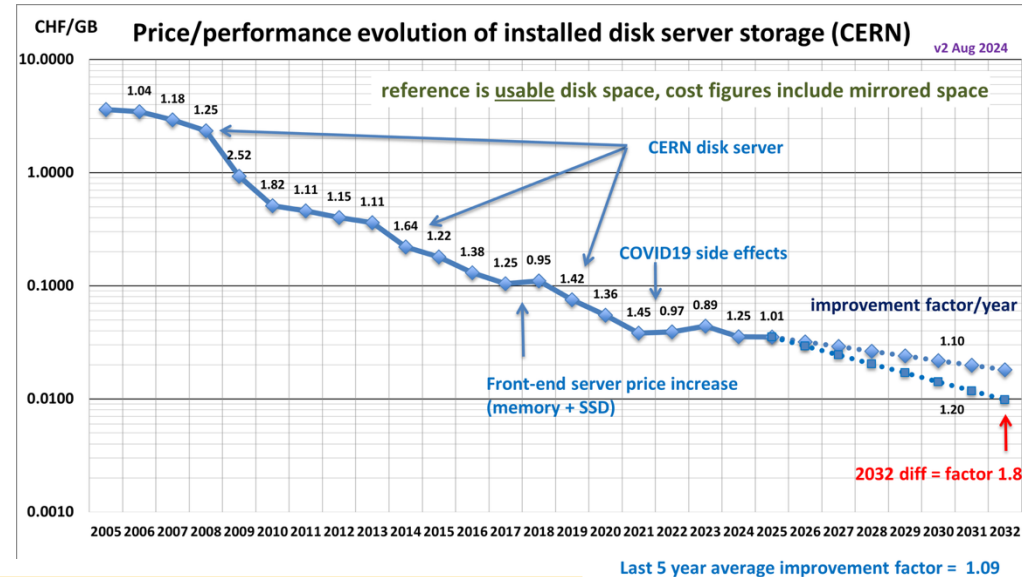
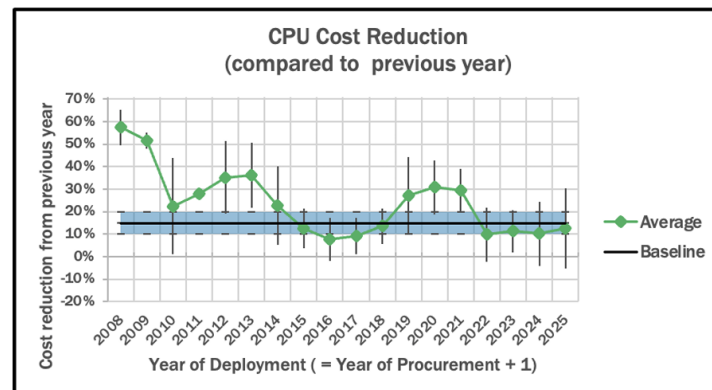
Multiple
projections
(lines): depend
on (successful)
R&D

In the meantime, technology ...



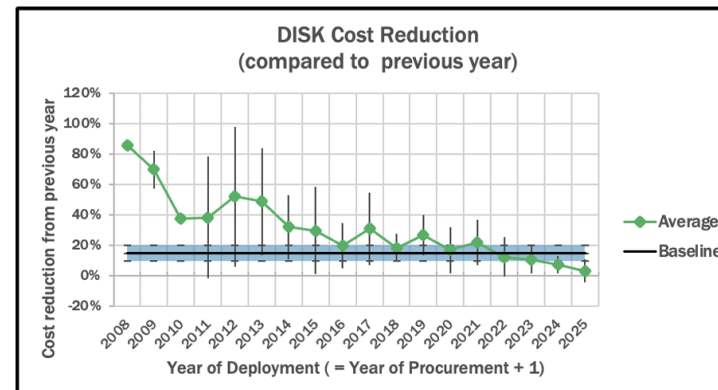
Average cost reduction for CPUs:

last 5 years: 15% - last 3 years: 11% - last year: 13%



Average cost reduction for disk:

last 5 years: 11% - last 3 years: 7% - last year: 3%



“Panzer/Sciabà”
plots

1 TB disk ~ 5 TB tape ~ 10
HS23 [1/2 CPU core]

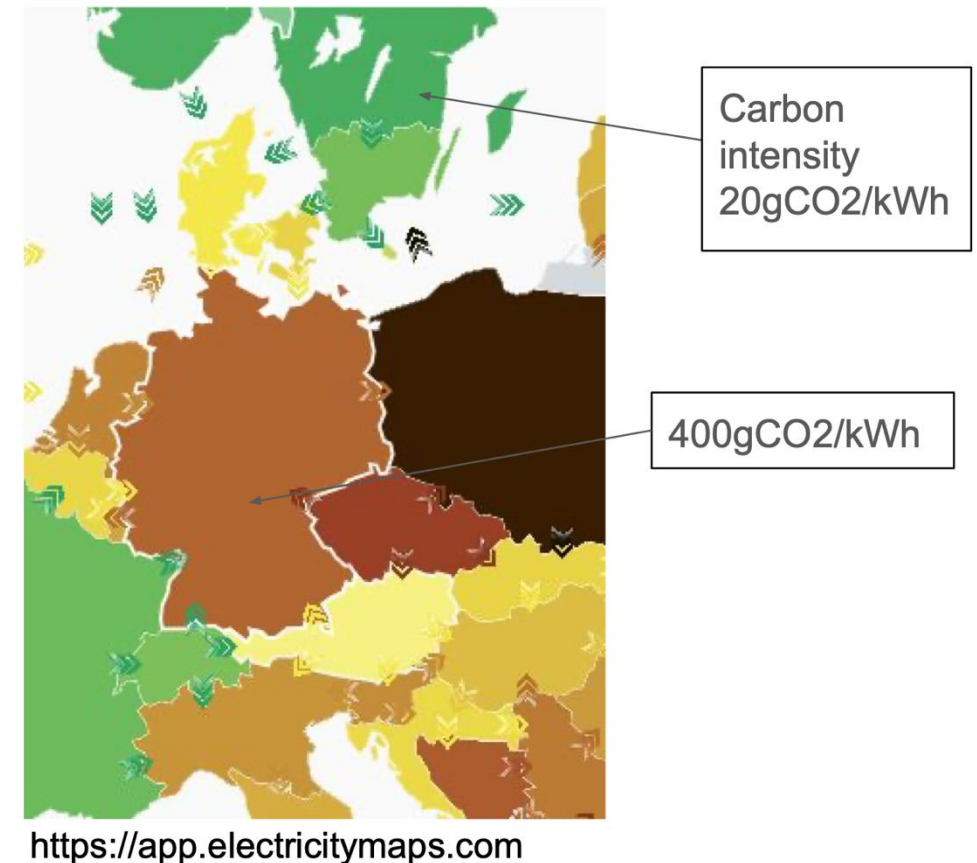
A “constant investment” on
computing buys X% more
resources every year.
X=10% → x2.5 in 10 years
X=20% → x6 in 10 years

Two questions ...

- Assuming we cannot get more money per year for computing, **where do we get the 12x-25x missing?**
- **Also, what is the environmental impact of HEP computing? Is it sustainable?**
- A non-exhaustive list
 - **Infrastructure** changes (where / how to get CPU and Disk, at which price)
 - **Technological** changes (use different technologies)
 - **Physics #1**: change analysis model (do the same physics with less resources)
 - **Physics #2**: reduce the physics reach (for example increasing trigger thresholds)
 - Not even considered here ... it is the “desperation move” if we fail with everything else
 - Use “**modern weapons**” (new/faster algorithms/tools)
 - Something **unexpected**...

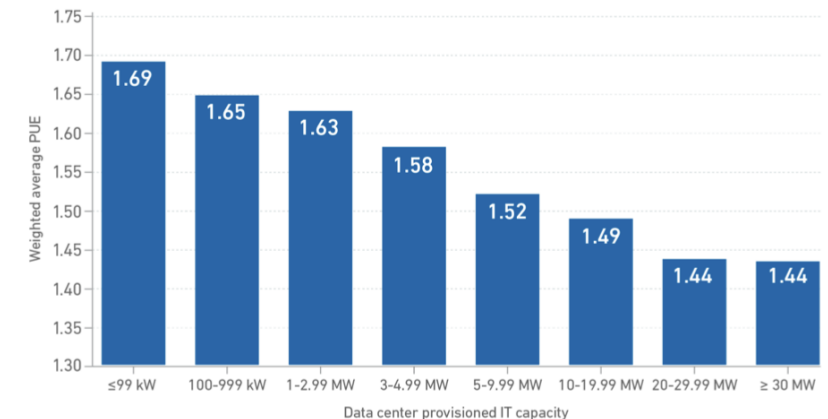
Environmental Sustainability

- Data centers and computing contribute 2-4% of global green house gas emissions, only expected to grow.
- Great variation of electricity emissions across countries and even regions.
- Can we be smarter about how we use existing facilities?
 - expose and use information on specific carbon impact
 - schedule workloads to run when electricity is cheaper/cleaner
- Consider carbon impact as an element of computing “performance” in benchmarking



Datacenters

- Climate-controlled building with enough electrical and cooling power for all the hardware
- Large rooms with racks
 - CPU boxes, HDD/SDD storage boxes, networking equipment, tape libraries, GPU boxes
- Cooling: Forced Air or Water
 - Newest developments: immersion cooling
- Power Usage Efficiency (PuE)
 - [Wikipedia](#): "PUE is a ratio that describes how efficiently a computer data center uses energy; specifically, how much energy is used by the computing equipment (in contrast to cooling and other overhead that supports the equipment)."
- Average PuE: 1.4-1.7 (between 40% and 70% is "wasted!")
- Large datacenters tend to have better PuE



(n=558)

UPTIME INSTITUTE GLOBAL SURVEY OF IT AND DATA CENTER MANAGERS 2023

uptime
INTELLIGENCE

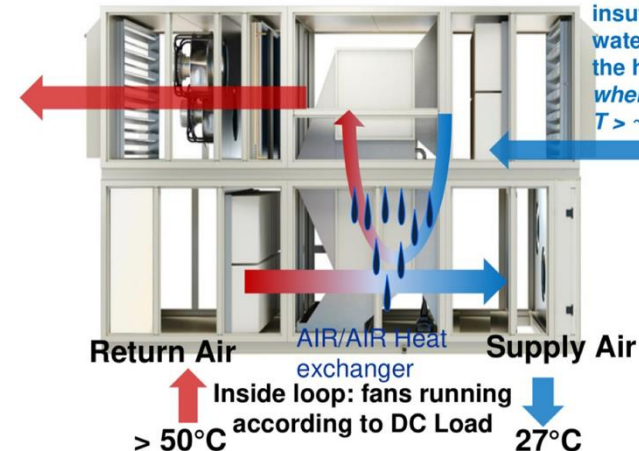
<https://journal.uptimeinstitute.com/large-data-centers-are-mostly-more-efficient-analysis-confirms/>

Towards high-efficiency data centers

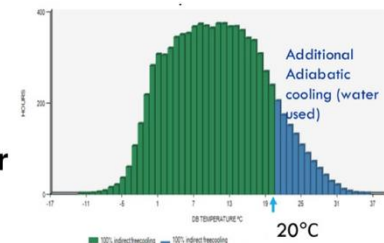
- Perlmutter Supercomputer @ NERSC (Berkeley Lab, US)
 - Direct water cooling, **PuE: 1.05-1.08**
- Green IT Cube: supercomputing center for GSI and FAIR (Darmstadt, D)
 - water cooling in doors of computer cabinets, **PuE: < 1.07**
- LHCb online farm (CERN)
 - Free air cooling, **PuE: < 1.1**
- Energy efficient data centers are coming, but not everywhere and not fast enough!



Outside Fans running 0-20% - 70%



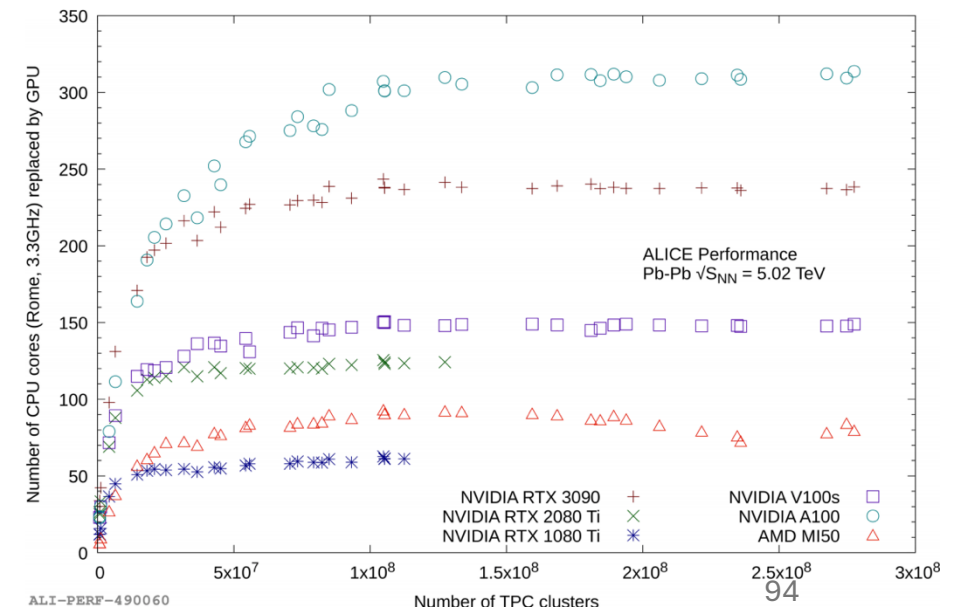
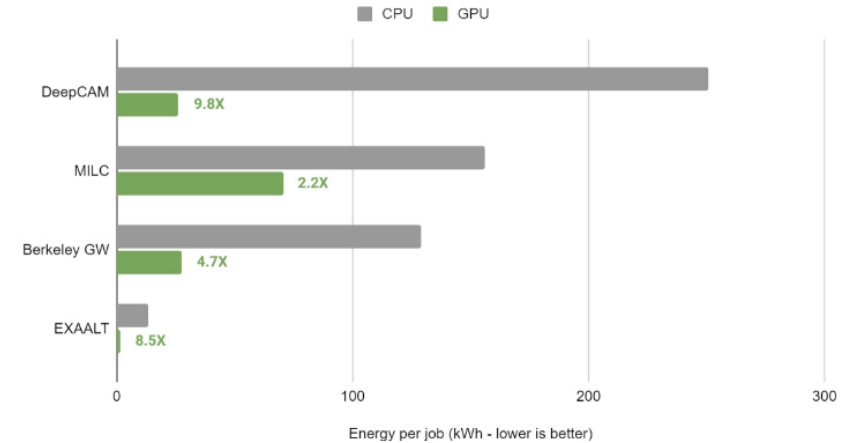
If outside airflow insufficient not enough, water is sprinkled on the heat exchangers when outside air $T > \sim 20^\circ\text{C}$



Efficient compute architectures

- GPUs are much more energy efficient
 - Perlmutter @ NERSC: 5x on average, up to 9.8x in weather forecast
- GPUs are much more compute-efficient
 - Up to 300 CPU cores replaced by a single GPU in ALICE's track reconstruction

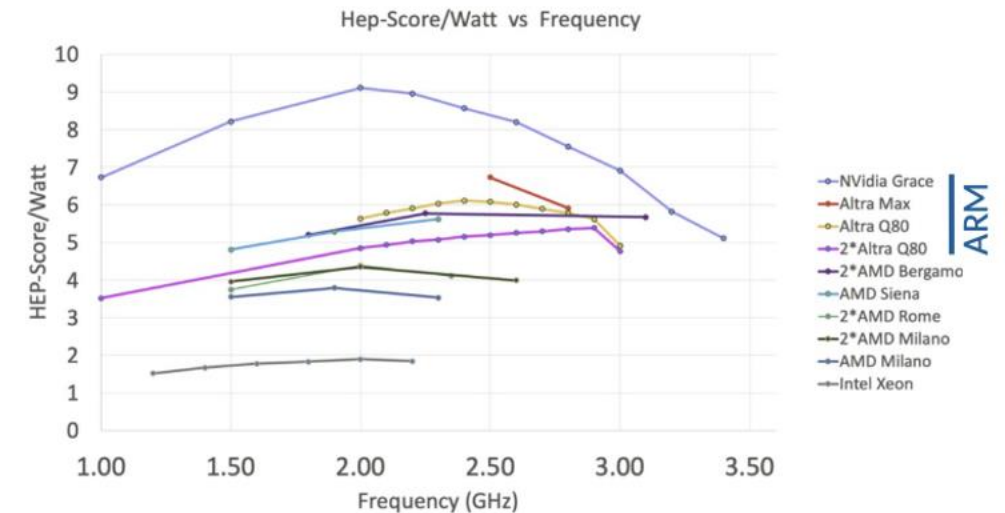
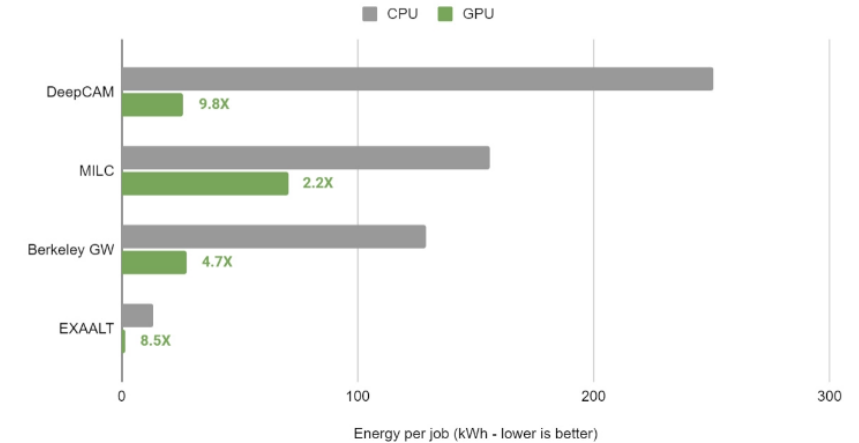
Energy Consumed per Job



Efficient compute architectures

- GPUs are much more energy efficient
 - Perlmutter @ NERSC: 5x on average, up to 9.8x in weather forecast
- GPUs are much more compute-efficient
 - Up to 300 CPU cores replaced by a single GPU in ALICE's track reconstruction
- Other architectures being considered
 - Mobile (low power) processors (ARM)
 - Code-in-hardware (“FPGA”, “ASIC”, ...)

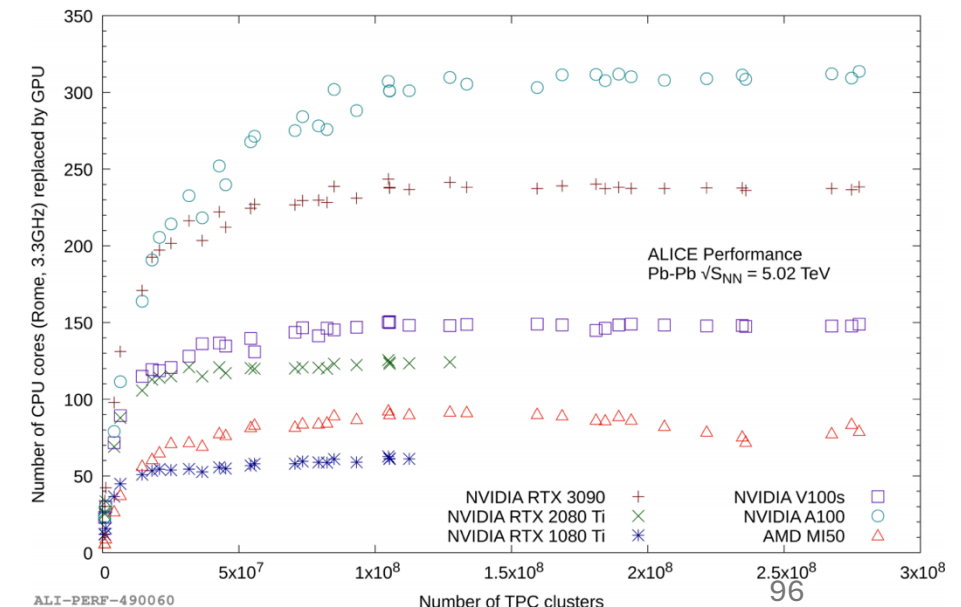
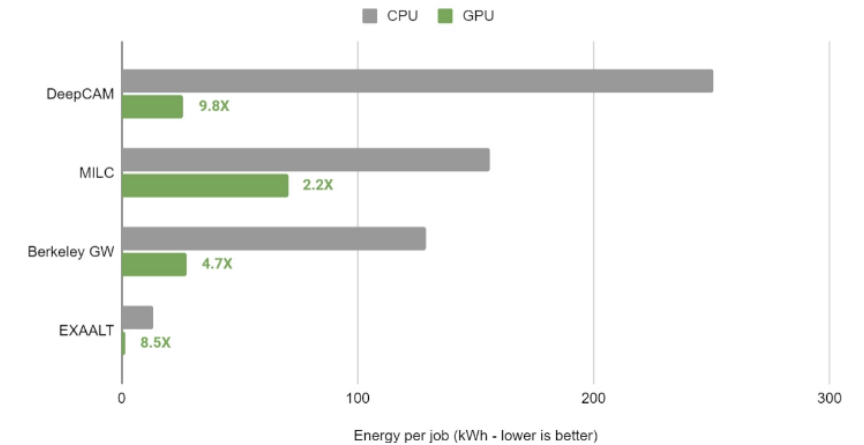
Energy Consumed per Job



Efficient compute architectures

- Large HPCs are and will be installing GPUs to boost their compute power and consume less electricity
- Can we use them?
 - Not easily - limited to mission critical algorithms
 - GPU programming is different: Need to use special code constructs (Essentially, if-statements have to be rethought)
 - We need frameworks to embrace Heterogeneous Computing
 - We need a way not to write the code once per platform
 - Portability libraries (Kokkos, Alpaka, OneAPI,...) allow to write algorithms once and then compile/execute on GPUs of different vendors and CPUs

Energy Consumed per Job

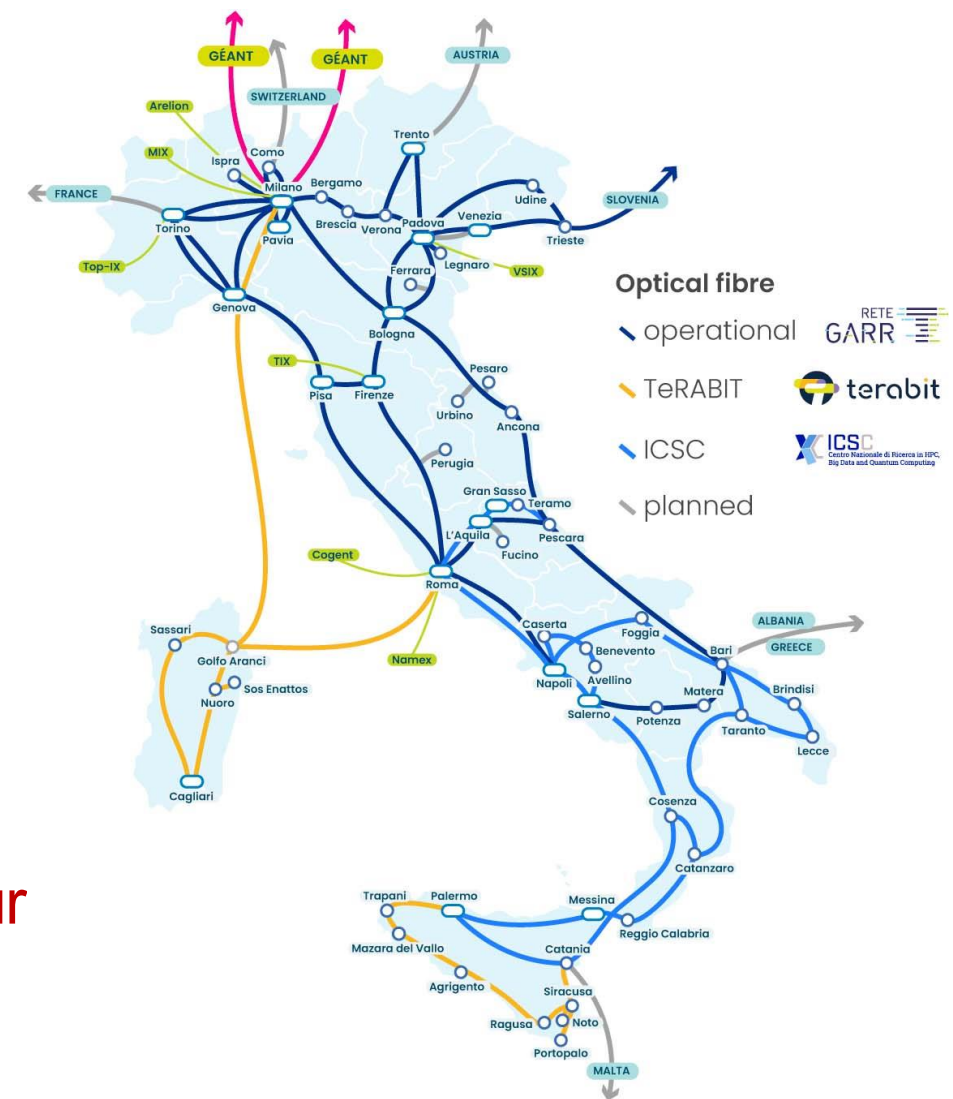


Infrastructure changes

- Today's HEP computing
 - Owned centers, long lifetime (10+ y)
 - Well balanced in storage vs CPU
 - FAs pay for resources + infrastructure + personnel

Is it the most economic/sustainable computing available today?

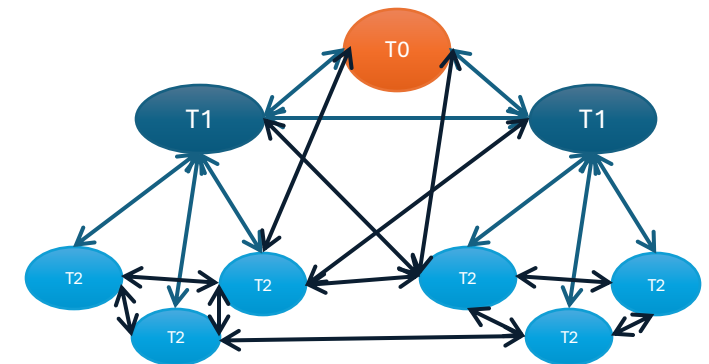
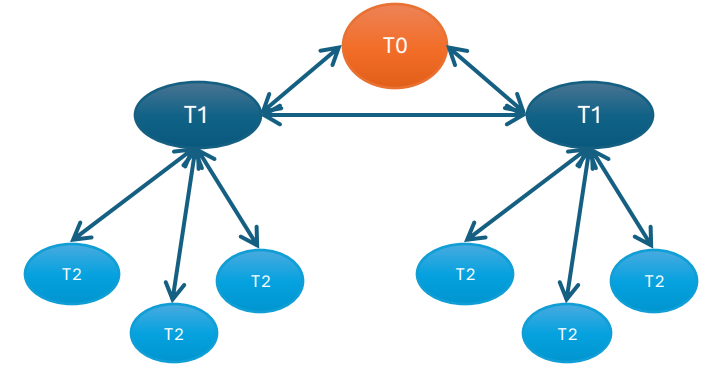
- **YES**, if you care about your data safety (and your capability to access it)
- **NO**, if you can use stateless resources (CPUs!)
 - They come and go fast
 - You can hire them (from a commercial provider, ...)
 - You can use “someone else” resources



“CPU for free can be found,
Disk for free cannot!”

Real operation mode today

- **Netflix, Spotify, ...** → commercial commodity networks available at a lower price / larger bandwidth than expected
- No need to have strict hierarchical network paths, → full mesh: every site can transfer from any other

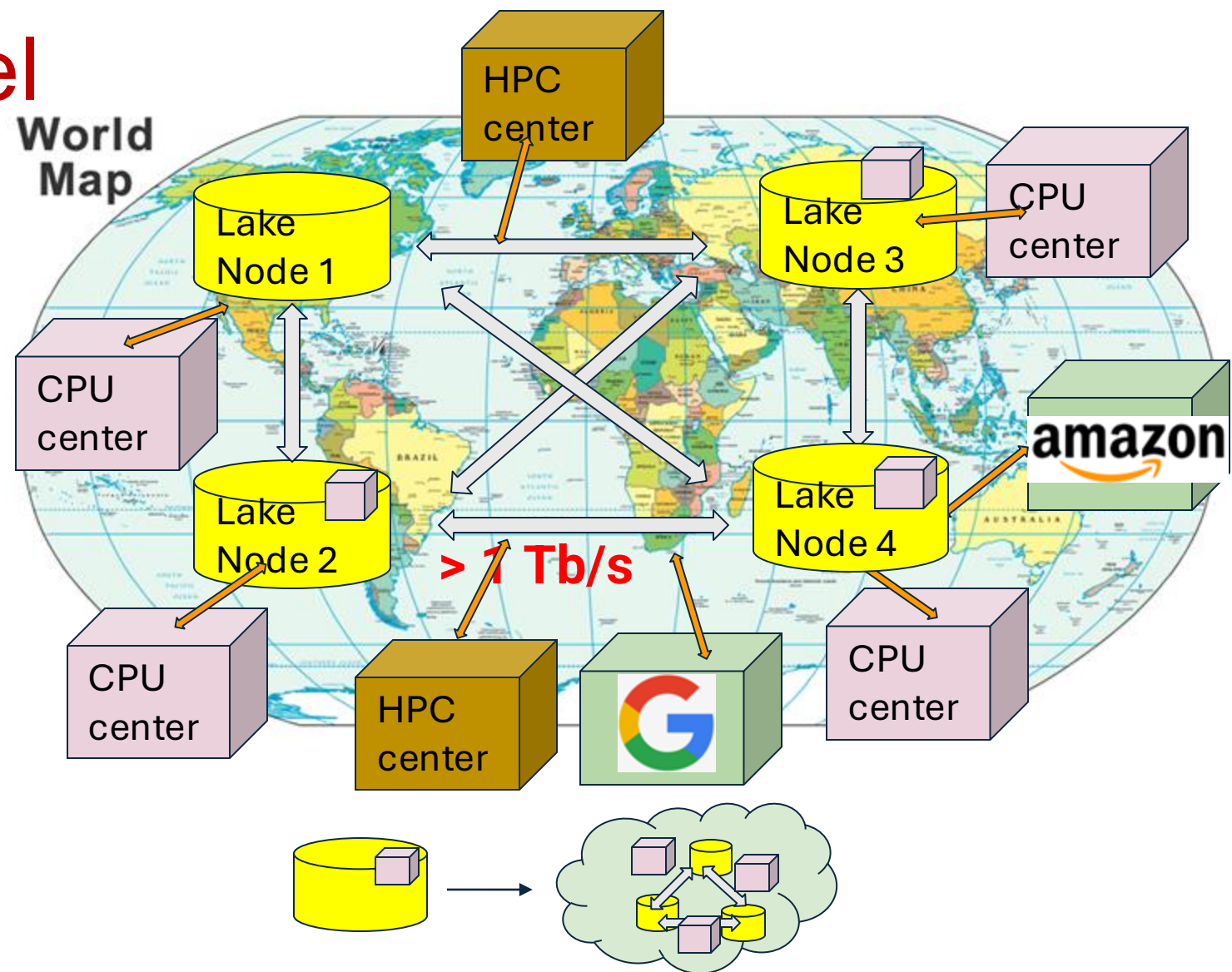


How to use the new network capabilities?

- **Direct Remote data access (a.k.a Streaming!)**
- You remember the problem with Data Driven: jobs go where data is
 - If a site has spare CPUs, but no data → not used
 - If a site has data, but no spare CPUs → jobs kept waiting
- If we remove the constraint of Data locality, match-making becomes very easy + efficient
 - Direct Remote Data Access: think of Youtube/Netflix!
 - You do not download the file, you access it over the network

The data lake model

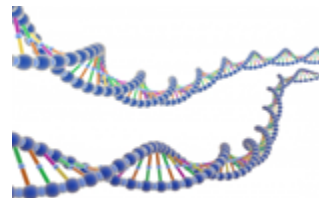
- Keep the real value from the experiments safe
 - (RAW) **data** and a solid baseline of **CPU** in owned and stable sites
 - Allow for multiple CPU resources to join, even temporarily
 - Eventually choosing the cheapest at any moment
 - Solid networking: use caches / streaming to access data
- Reduce requirements for Computing resources
 - Commercial Clouds
 - Other sciences' resources
 - SKA, CTA, Dune, Genomics, ...
 - HPC systems



ProtoDune 2-3
GB/s (like CMS);
Real Dune 80x



SKA up to 2
PB/day



A single
genome ~
100 GB. a 1M
survey = 100
PB



CTA projects
to 10 PB/y
100

Commercial clouds

- Massive data centers with \$B investment provide access to vast amounts of resources
- HEP resources are sizable but tiny compared to industry
- Industry selling compute in small slices for profit
- Allows for both large scale (if you can pay for it) and fine granularity
- Some hyperscalers (Google) offer subscription models that allow to boost into unused capacity
- In general, higher prices to buy elasticity

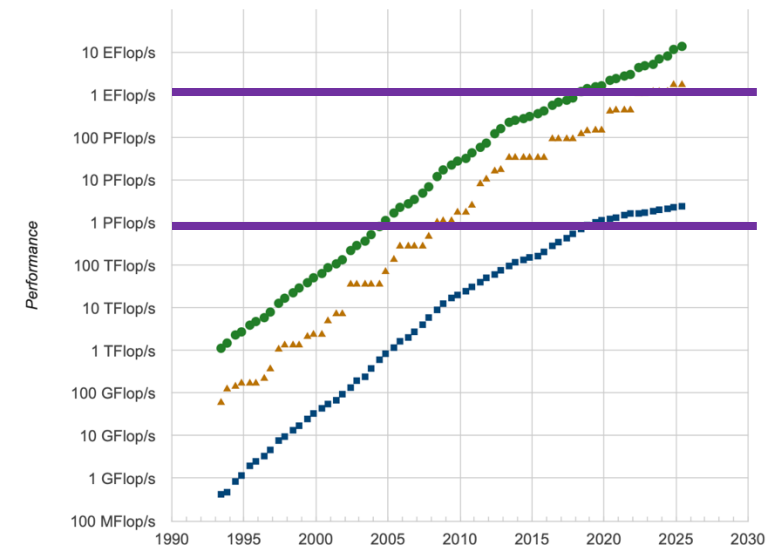
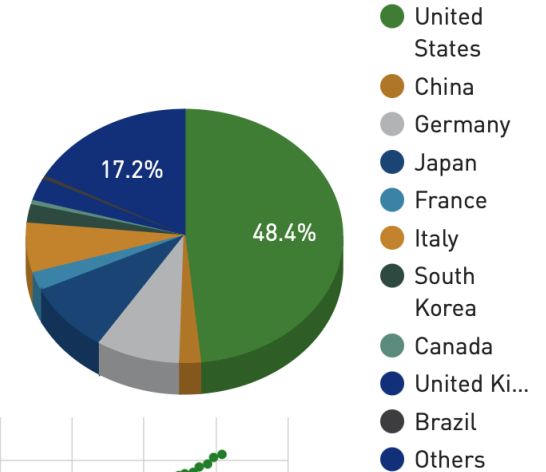


Supercomputing (HPC)



- High Performance Computing (HPC) is designed for single large applications using significant resources
 - Scientific use cases: climate models, lattice QCD
 - specialized hardware with very fast interconnects
- Recently they are opening up to HEP workflows (HEP = high throughput computing (HTC))
 - Even we don't really need fast interconnects

Countries Performance Share



1 Exaflop

1 Petaflop

1 Petaflop = 10^{15} floating point operations per second
1 Exaflop = 10^{18} floating point operations per second

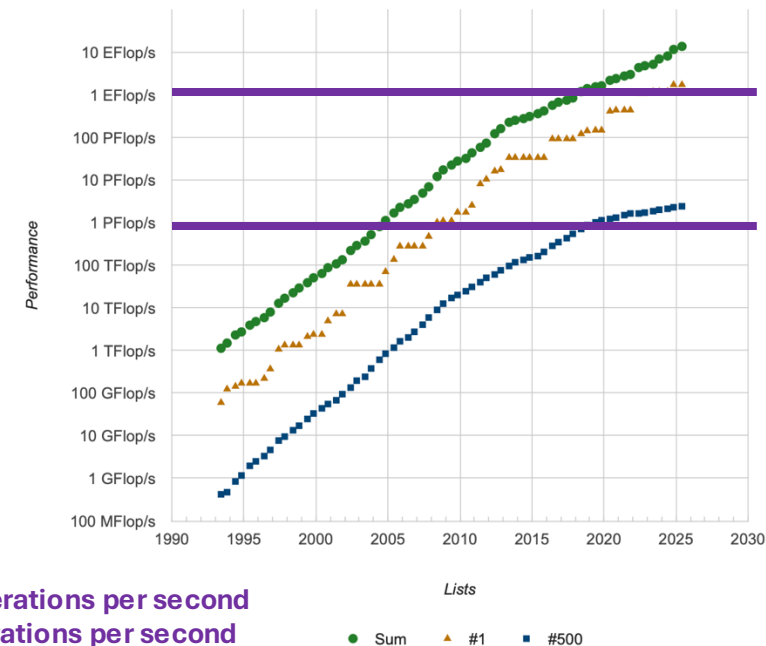
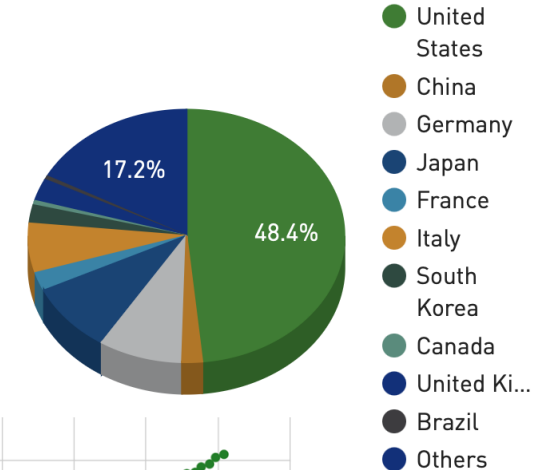
Supercomputing (HPC)

Many non-trivial problems to solve

- Data access (access, bandwidth, ...)
- Accelerator Technology (GPU, FPGA, TPU, ...)
- Submission of tasks (MPI vs Batch systems vs proprietary systems)
- Node configuration (low RAM/Disk, ...)
- Not-too-open environment (OS, ...)
- Processing time is allocated through approval processes based on science use case
- Resources are not necessarily available 24/7/365



Countries Performance Share



1 Exaflop

1 Petaflop

1 Petaflop = 10^{15} floating point operations per second
1 Exaflop = 10^{18} floating point operations per second

Physics #1: change analysis model

Most HEP physics analysis use a sequential model

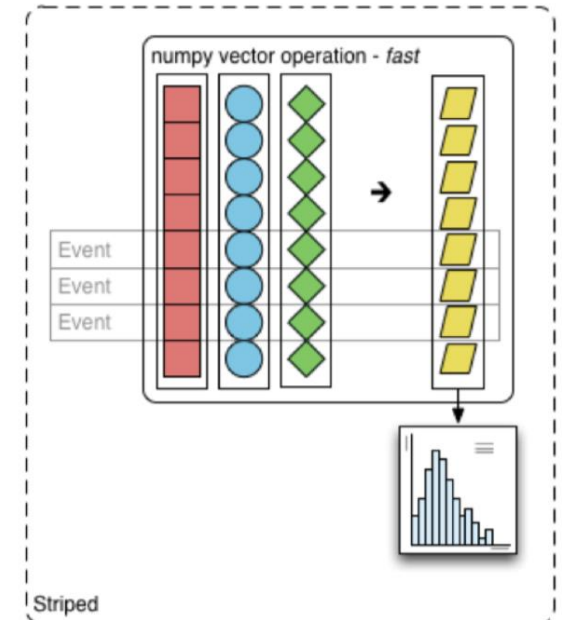
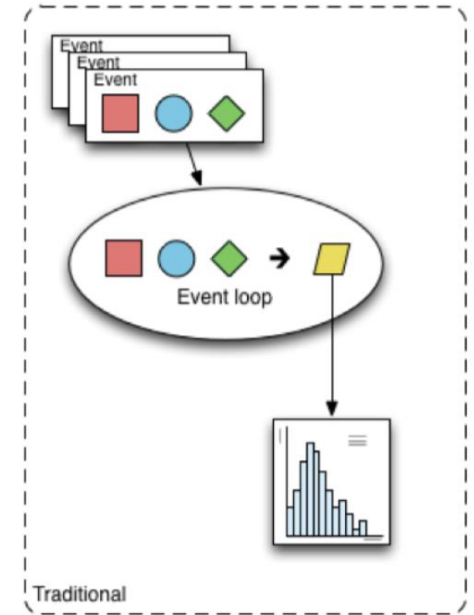
«event loop» on a single CPU:

- Load relevant values for a specific event into local variables
- Evaluate several expressions
- Store derived values
- Repeat (explicit outer loop)
- Make it faster by making it embarrassly parallel using a lot of CPUs (for example, using the GRID)

Big data tools are known to be better at this

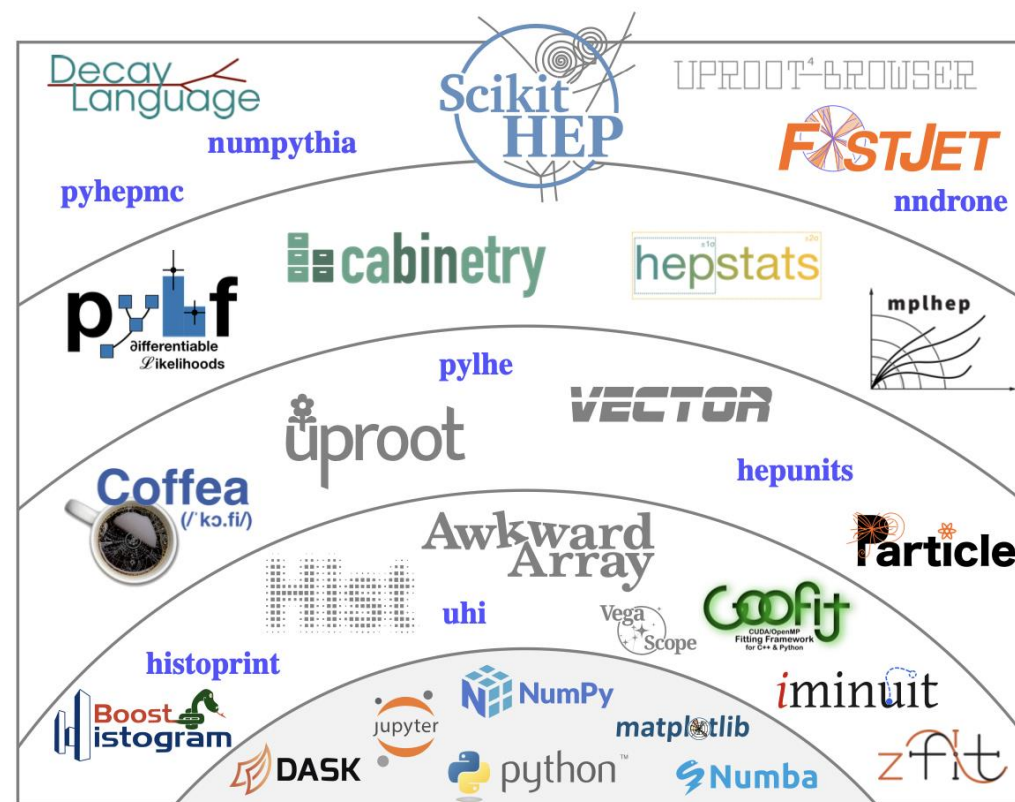
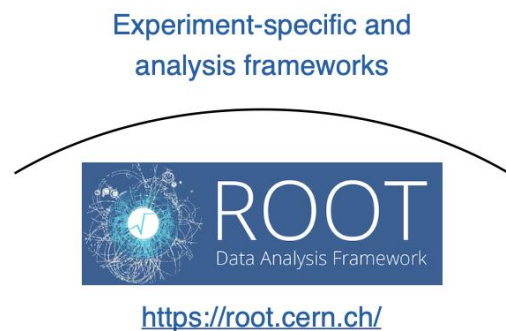
Columnar analysis:

- Load relevant values for many events into contiguous arrays
- Evaluate several array programming expressions
- Implicit inner loops
- Store derived values



Physics #1: change analysis model

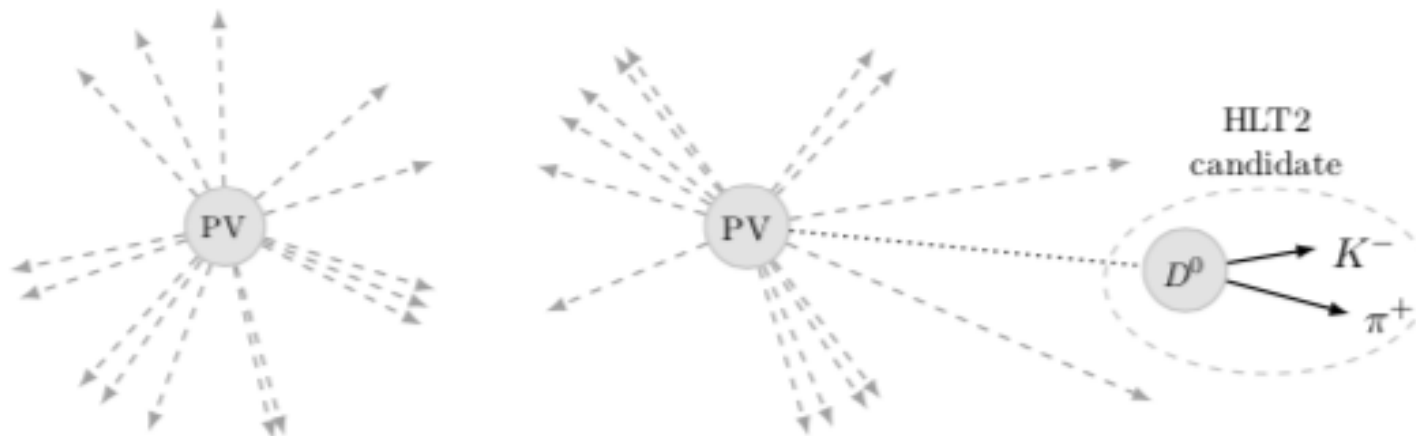
From vertically-integrated solution to ecosystem



Physics #1: reduce storage footprint



- **Selective persistency:** write out only the “interesting” part of the event.



- **Turbo stream:**
 - Minimum output: only HLT2 signal candidates

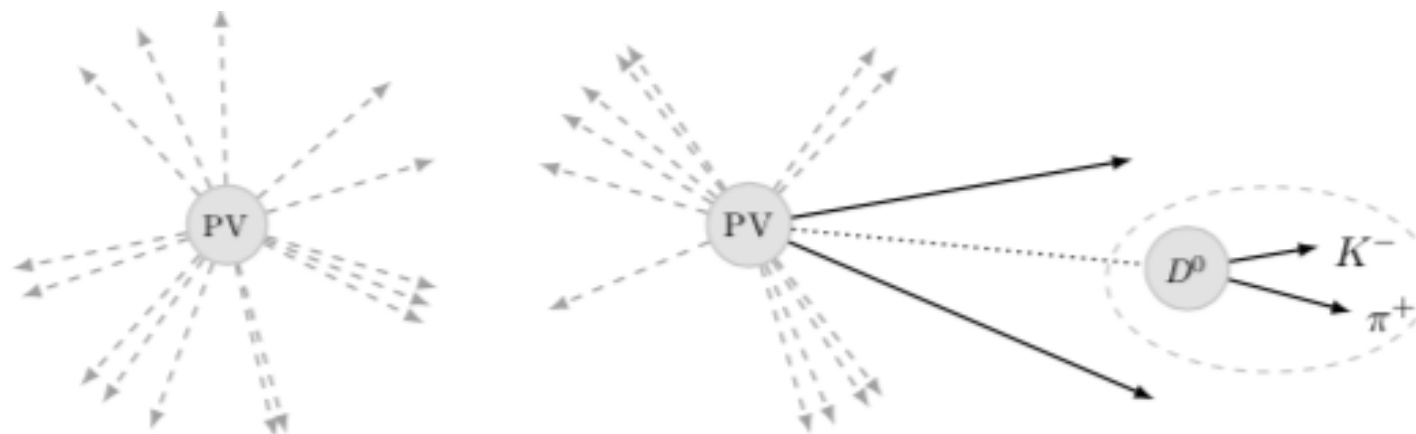
Limitations: cannot refit tracks and PVs offline, rerun flavour tagging etc.

Advantage: Event size $O(10)$ smaller than RAW

Physics #1: reduce storage footprint



- **Selective persistency:** write out only the “interesting” part of the event.



- **Turbo stream:**
 - Minimum output: only HLT2 signal candidates
 - Optionally: (parts of) pp vertex (e.g. “cone” around candidate for spectroscopy searches)

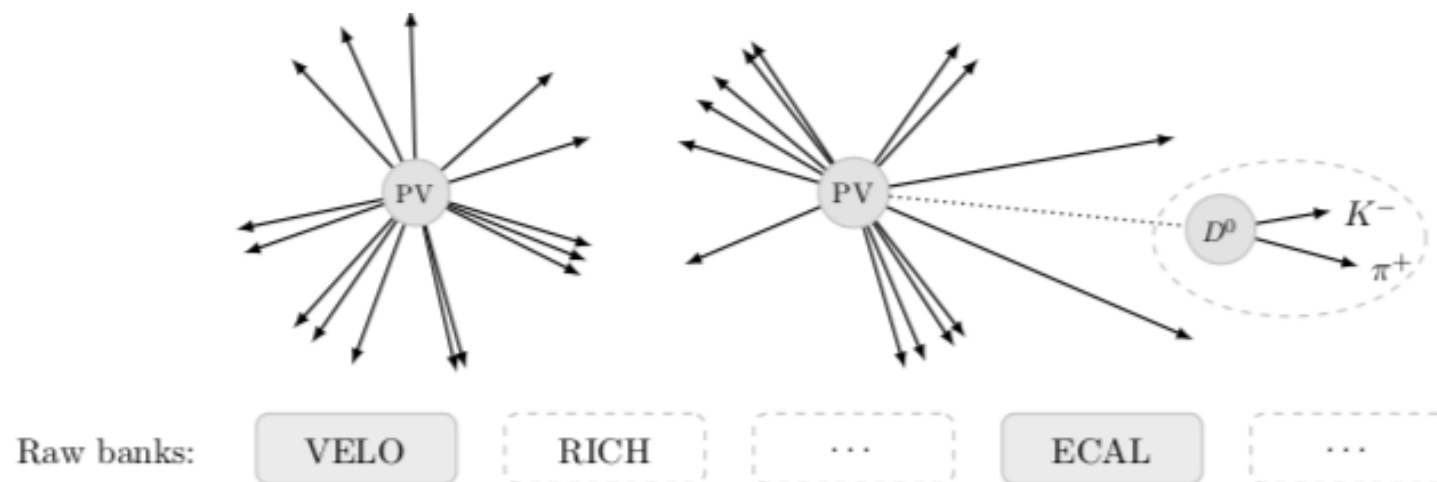
Limitations: cannot refit tracks and PVs offline, rerun flavour tagging etc.

Advantage: Event size $O(10)$ smaller than RAW

Physics #1: reduce storage footprint



- **Selective persistency:** write out only the “interesting” part of the event.



- **Turbo stream:**
 - Minimum output: only HLT2 signal candidates
 - Optionally: (parts of) pp vertex (e.g. “cone” around candidate for spectroscopy searches)

Limitations: cannot refit tracks and PVs offline, rerun flavour tagging etc.

Advantage: Event size $O(10)$ smaller than RAW

- **FULL stream:** all reconstructed objects in the event
 - Optionally adding selected RAW banks

Physics #1: reduce storage footprint

- CMS has developed more and more reduced data formats
- “nanoAOD” is the prevalent analysis format in CMS
- Event size reduced by a factor 3000x since the start of Run-1
- Note: only very high-level quantities are saved; not all analyses can use it
 - e.g. flavour physics analyses

| Data Tier | Size (kB) |
|---------------------|--------------------|
| RAW | 1000 |
| GEN | < 50 |
| SIM | 1000 |
| DIGI | 3000 |
| RECO(SIM) - 2010 | 3000 |
| AOD(SIM) - 2012 | 400 (8x reduction) |
| MINIAOD(SIM) - 2015 | 50 (8x reduction) |
| NANOAOD(SIM) - 2018 | 1 (50x reduction) |

Use “modern weapons”

- These can be from the technology point of view (Big Data Tools)...
- ... or novel ways to write algorithms.
- AI in general and Machine Learning / Deep Learning techniques obviously stand up
- The space / time here is way too short to go into any detail, but by now ML techniques are used everywhere in HEP processing
 - Trigger level (even on FPGA)
 - Simulation (GAN tools are very promising)
 - Reconstruction (... everywhere, from S/N separation to clustering in calorimeters and trackers)
 - Analysis (selection, interpretation, ...)

The AI/ML zoo

- **Fully Connected Neural Networks (FCNNs / MLPs)**
 - Used in early applications (e.g. event classification, regression)
 - Still widely used for tasks with structured tabular input (e.g. particle 4-vectors)
 - *Examples: S/B discrimination, parameter estimation, ..*
- **Convolutional Neural Networks (CNNs)**
 - Suitable for image-like data: calorimeter hits, tracking detector layouts, jet images
 - Benefit from local connectivity and translational invariance
 - *Examples: jet tagging, energy deposition maps, neutrino detectors*
- **Graph Neural Networks (GNNs)**
 - Represent events as graphs (e.g., hits, tracks, or particle interactions as nodes/edges)
 - State-of-the-art for tracking, jet reconstruction, and physics object identification
 - *Examples: Track finding, calorimeter clustering, particle flow*
- **Autoencoders (VAEs) (and Variational-AE)**
 - Used for anomaly detection and dimensionality reduction
 - *Examples: Searching for rare or unknown physics events.*

The AI/ML zoo

- **Generative Adversarial Networks (GANs)**
 - Fast surrogate models for simulation (e.g., calorimeter shower generation).
 - *Examples: Simulation acceleration, anomaly detection*
- **Transformers**
 - Originally from Natural Language Processing (NLP), now extended to handle structured or variable-length (long!) inputs.
 - Strong performance in classification and generative modelling, even in physics.
 - *Examples: Event classification, generative modelling, scientific document parsing.*
- **Diffusion models**
 - Model data generation as reversing a diffusion process (progressive noise addition)
 - *Examples: fast calorimeter and tracking simulation, anomaly detection, structured generation*

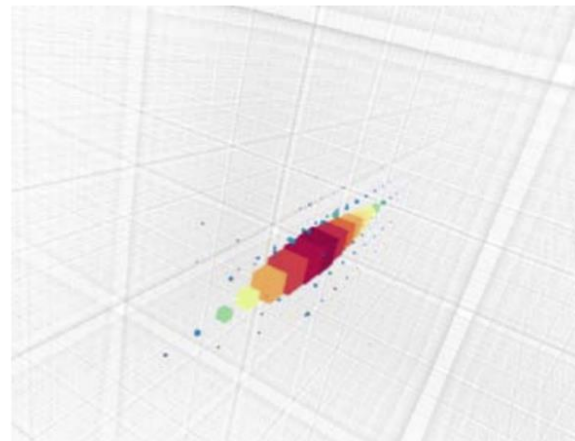
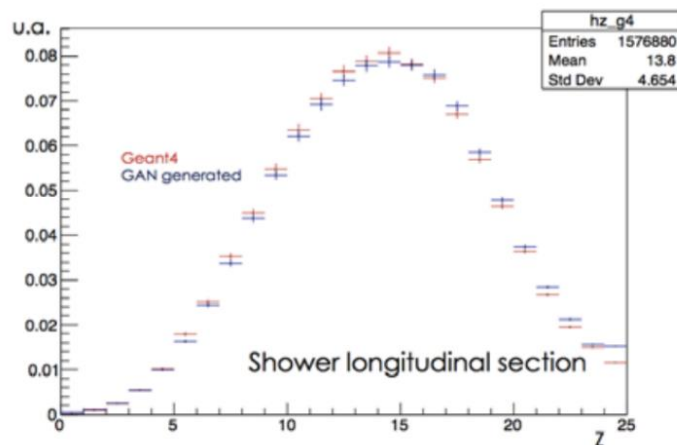
ML usage patterns #1

- At **trigger level**, modern tools ([hls4ml](#), [BM](#), [LeFlow](#), ...) allow to write on FPGA the result from the training on “largish” machine learning networks, taking into account pruning to match the limited resources
- Applications under study
 - Bkg and trigger rate reduction
 - Signal specific trigger paths
 - Anomaly detection in data taking
 - Unsupervised new physics mining
- Existing implementations, e.g. LHCb HLT selections in Run3
- Next-generation trigger systems → real-time reconstruction → real time analysis
- Challenge is the trade-off between algorithmic complexity and the performances achievable under severe time constraints in inference



ML usage patterns #2

- The production of simulated events is extremely intense from the computation standpoint
 - up to the point it might impact the physics reach of the experiments
- **ML can help to reduce such load**
 - Calorimeter shower surrogate simulator
 - Analysis-level simulator
 - Pile-up overlay generator
 - Monte Carlo integration
 - ML-enabled fast-simulation
- As an example, **GANs** have shown the potential to mimic more complex iterative algorithms (like those in Geant4) with a huge gain in timing



Longitudinal shower shape in a calorimeter from 100 GeV e^- from [here](#). Timing is 1 minute vs 0.04 msec

ML Usage patterns #2

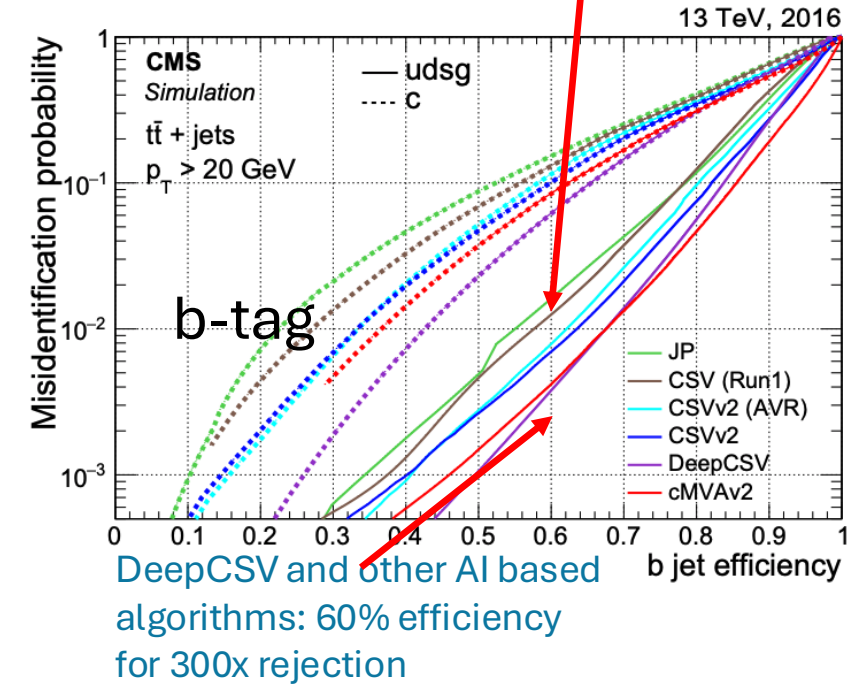
Online/offline **reconstruction** might be partially replaced by ML surrogate models (approximate → faster) or new algorithms (offering unprecedented performance) might partially replace existing algorithms.

- Charged particle tracking (GraphNN, vertexing, ...)
- Calorimeter reconstruction (local, clustering, ...)
- Particle flow (GraphNN, ...)
- Particle identification (boosted
- jets, isolation, ...)
- Pileup mitigation

ML Usage patterns #2

A couple of examples on how ML is used at
reconstruction level

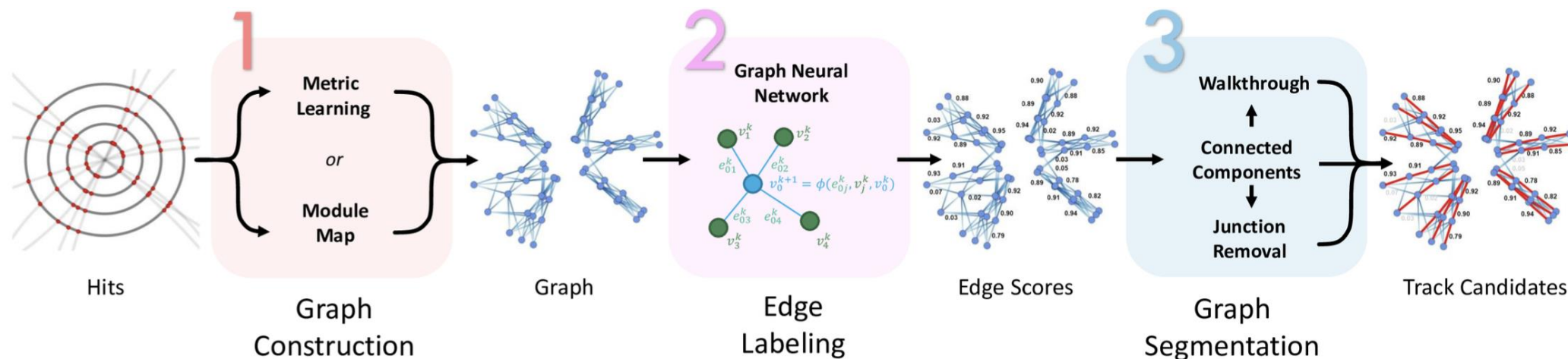
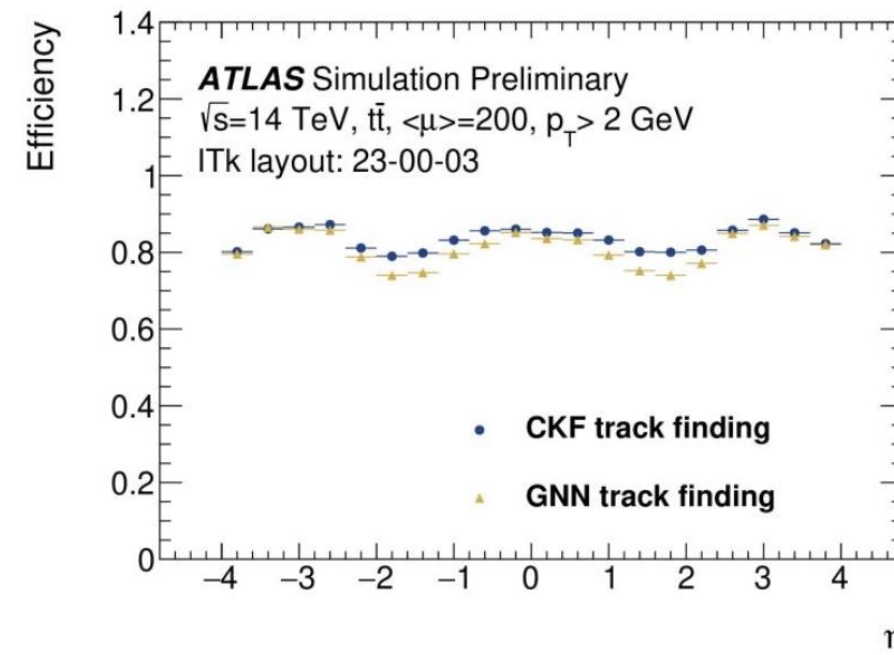
- **Improvement in classification** (S vs B, and in general category A vs B, C, ...) using a large number of (even poorly) discriminating variables



ML Usage patterns #2

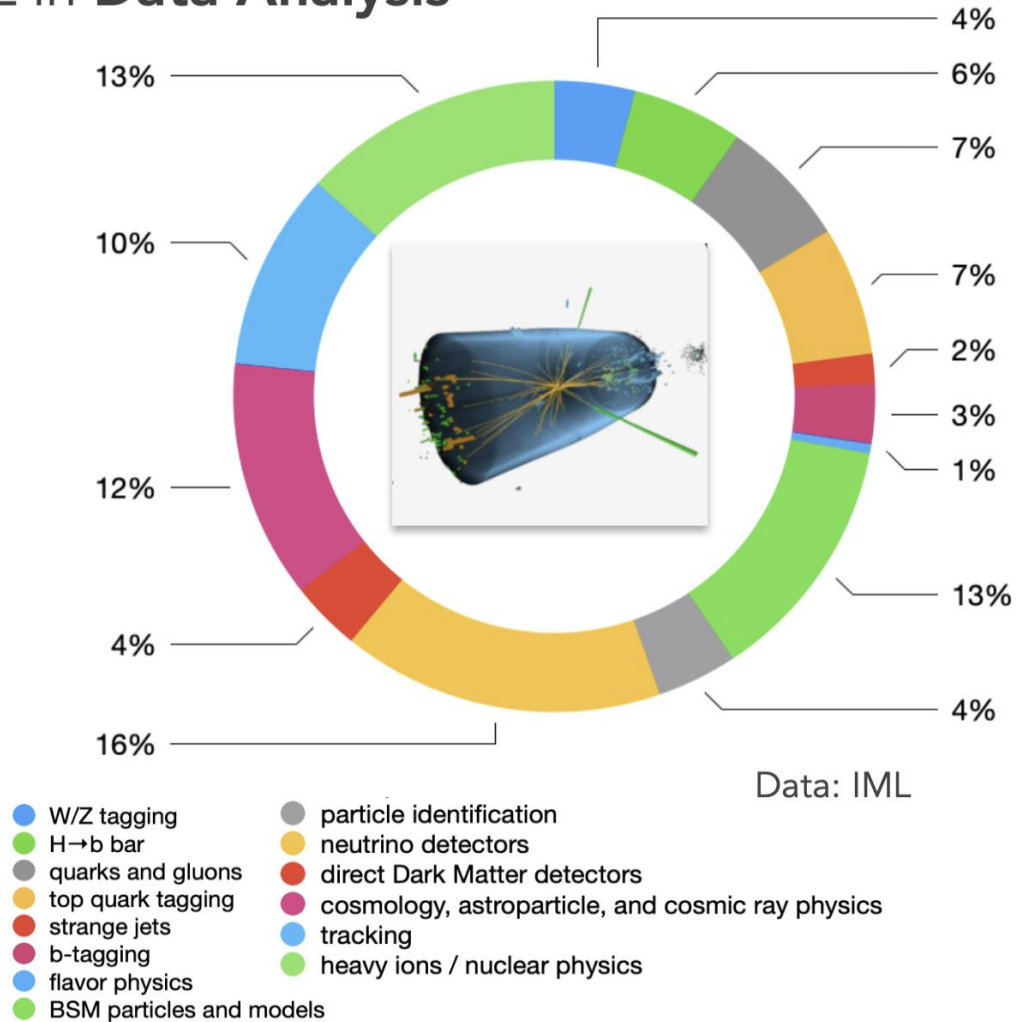
A couple of examples on how ML is used at **reconstruction level**

- **Clustering algorithms** which exhibit combinatorial explosion with classical algorithms (jet clustering, tracking)
 - CNNs (input-as-images), Graph Networks



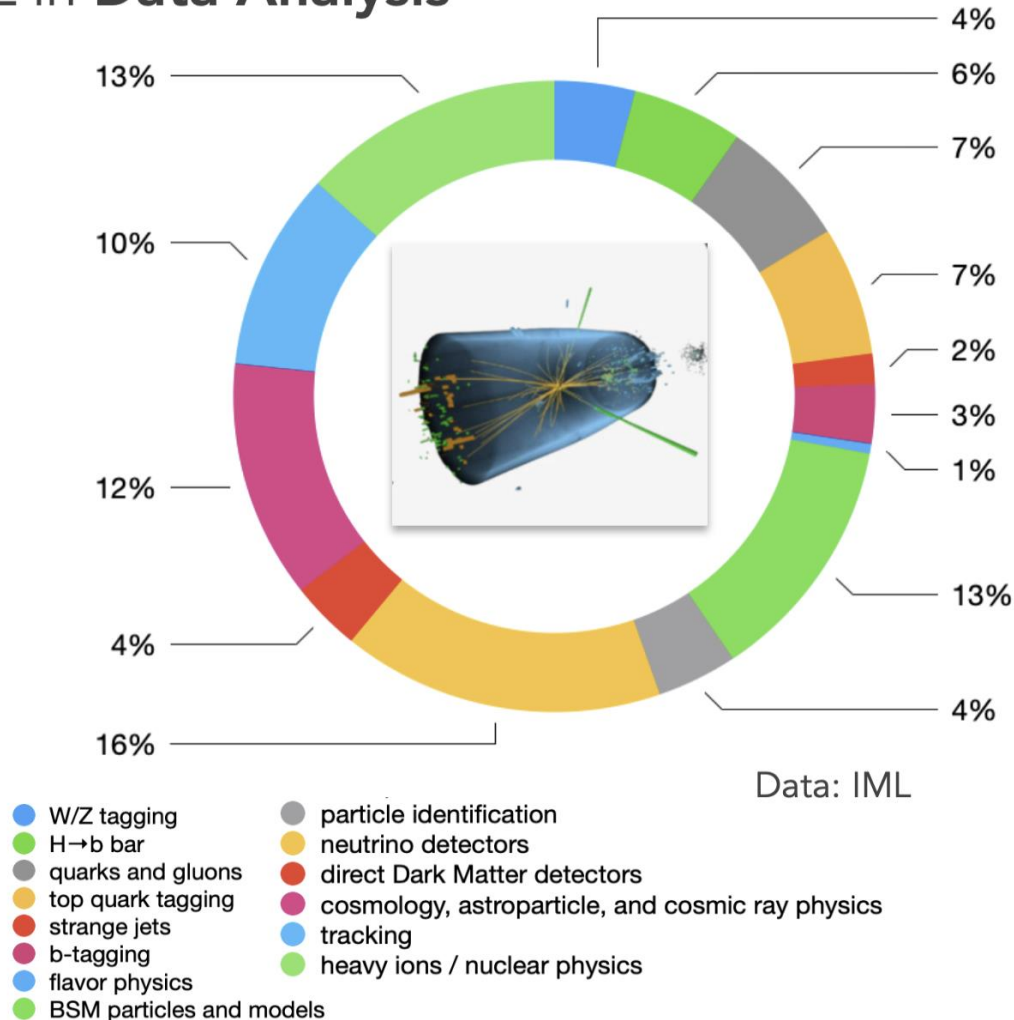
ML usage patterns #3, #4

ML in Data Analysis



ML usage patterns #3, #4

ML in Data Analysis



ML in Computing Operations

Application of ML to **non-event (meta-)data** might help to increase efficiency and reduce the need of personpower in Ops, e.g. automating specific tasks, developing intelligent/adaptive systems, ultimately acting on the full chain - from data collection to data analysis - and make it more agile

- Detector control
- Data quality monitoring
- Operational intelligence
- Predictive maintenance



ML usage patterns #5



“ChATLAS”: A prototype LLM-based AI-assistant in a

- Data gathering part is interesting
 - Docs: twiki (>2k), sw docs (>500), e-groups/emails archive (>10k), indico meetings’ agendas incl. attached slides and minutes (>440k), Mattermost, Jira tickets, experiment’ papers and internal notes (>66k)
 - Either HTML or scraped into markdown
- Many open challenges:
 - highly heterogeneous data
 - ensure that collaboration DBs are accessible and exportable; websites should live on a git repo; pubs should be saved as latex, and compiled separately; discussion forums should have anonymisation options... Estimates indicate that this would have saved ~1 yr of data wrangling
 - Hallucinations are a real problem
 - Not many gpu-hrs, but many expert-hrs, needed for any high-quality fine-tuned AI assistant

ML usage patterns #5



What is the crack veto for electron reconstruction?

The crack veto for electron reconstruction is applied to ensure that no electrons are reconstructed in the crack region of the detector. Specifically, the crack veto is defined for electrons with a transverse momentum (p_T) greater than 10 GeV and is applied in the region where the absolute value of the pseudorapidity ($|\eta|$) is between 1.37 and 1.52. This is to avoid issues related to the detector's performance in that region.

In summary, the crack veto conditions are:

$$p_T > 10 \text{ GeV}$$

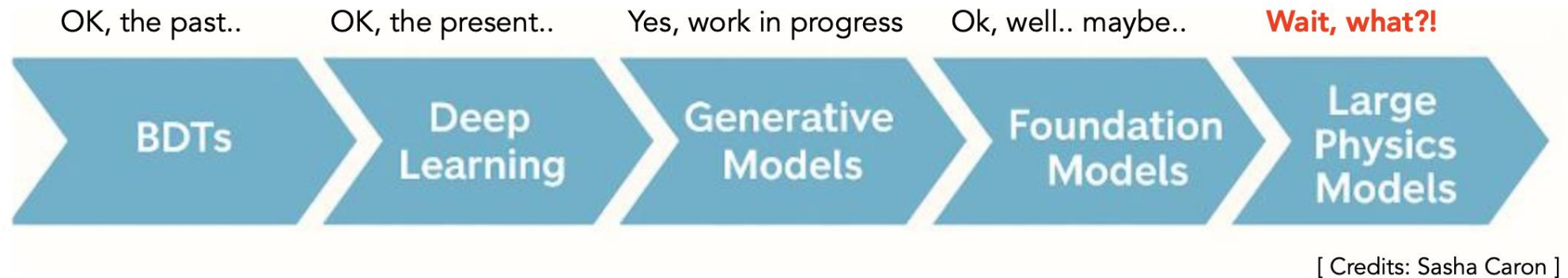
$$1.37 < |\eta| < 1.52$$

Are reconstructed tracks treated as massless in ATLAS MC?

In the ATLAS experiment, reconstructed tracks are typically treated as massless particles in the Monte Carlo (MC) simulations. This is because, in high-energy physics, the momentum and energy of particles are often more relevant than their mass, especially for particles like electrons, muons, and neutrinos, which are treated as massless in many analyses.

However, it's important to note that the actual mass of particles can be taken into account in specific analyses or when simulating certain processes. The treatment may vary depending on the specific requirements of the analysis being conducted. If you have a particular analysis or scenario in mind, please provide more details for a more tailored response.

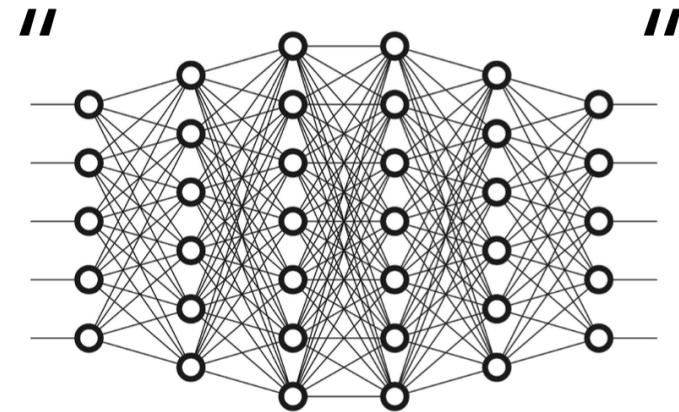
A possible AI / ML path



Is this a possible path that *extrapolates* the past to a “possible” future?

Foreseeing **“one big trainable unit”**, that just goes end to end, and we get rid of more and more of the traditional pipelines we are confident with? What if e.g. the entire physics data “analysis” pipeline becomes trainable, e.g. all the experiment code becomes an end-to-end differentiable pipeline that can be adjusted for a goal?

How will we treat data? Will a large foundation model be a black box, or will it be interpretable? If the latter, through human-in-the-loop? Will we want to talk to our data via large language models? What about performance gains? What about scientific rigour and reproducibility?



A Large "fundamental physics" foundation model?

A foundation model in general:

(D. Bonacorsi)

- A large-scale ML model trained on broad and diverse data, at scale, mainly with self-supervised learning objectives, designed to be adaptable to a wide range of downstream tasks with "minimal" fine-tuning
 - ◊ In general: training on text (e.g. web, papers), audio, video, code, images, math, structured data, ..

A foundation model for fundamental physics (**LPM**)?

- Training on large and diverse datasets within a given scientific domain
 - ◊ In HEP: detector-level raw data, simulation-level data, reco-level data, analysis-level papers/plots/logbooks/docs, metadata, ..
- + transfer learning (minimal fine-tuning) + many parameters + multipurpose + some capability not explicitly included during training..

Large Physics Models: Towards a collaborative approach with Large Language Models and Foundation Models

Kristian G. Barman^{*1}, Sascha Caron^{*2}, Emily Sullivan³, Henk W. de Regt⁴, Roberto Ruiz de Austri⁵, Mieke Boon⁶, Michael Färber⁷, Stefan Fröse⁸, Faegheh Hasibi⁹, Andreas Ipp¹⁰, Rukshak Kapoor¹¹, Gregor Kasieczka¹², Daniel Kostić¹³, Michael Krämer¹⁴, Tobias Golling¹⁵, Luis G. Lopez¹⁶, Jesus Marco¹⁷, Sydney Otten^{18,19}, Pawel Pawlowski¹, Pietro Vischia²⁰, Erik Weber¹, and Christoph Weniger²¹

PROs

- Tailored to physics tasks and structures
- Scaling to complex inference across simulation, data and theory
- Shared infrastructure → scientific collaboration at scale
- Potential to enhance discovery, reproducibility, and understanding
- Can be open, not in the hand of companies
- Prototype for other fields of science

CONs:

- High cost: compute, data, engineering, manpower, money
- Epistemic opacity: hard to interpret latent space reasoning
- risk of premature hype without careful testing
- risk of "dead of arrival" (obsolete before completion)
- risk of being less useful / capable

arXiv:2501.05382

Conclusions

- In this (long) walk I tried to show you how the complexity of Computing and Software systems for High Energy Physics has dramatically increased in the last ~30 years, becoming an integral part of the planning for new experiments, ... and their cost!
- In parallel, new skills and competencies have become more and more important. We now need more and more “physicists with CS skills”
- It is an interesting time to be in the Computing and Software for HEP
 - A complex task, no trivial solutions → we need new ideas
 - At the forefront of technology
 - Please join!

Acknowledgments

Thanks to

- Tommaso Boccali for his lecture at the 2021 CERN+FNAL HCP summer school and updates
- Paul Laycock for his lectures to the 2024 CERN summer students
- Arnulf Quandt for his lectures at the 2025 CERN school of computing
- Daniele Bonacorsi for his talk at the 2025 INFN-CCR workshop
- Ben Couturier for his seminar at Krakow university
- Oliver Gutsche for his lecture at the 2024 CERN+FNAL HCP summer school