



# COMPUTING MODELS IN HIGH ENERGY PHYSICS

---

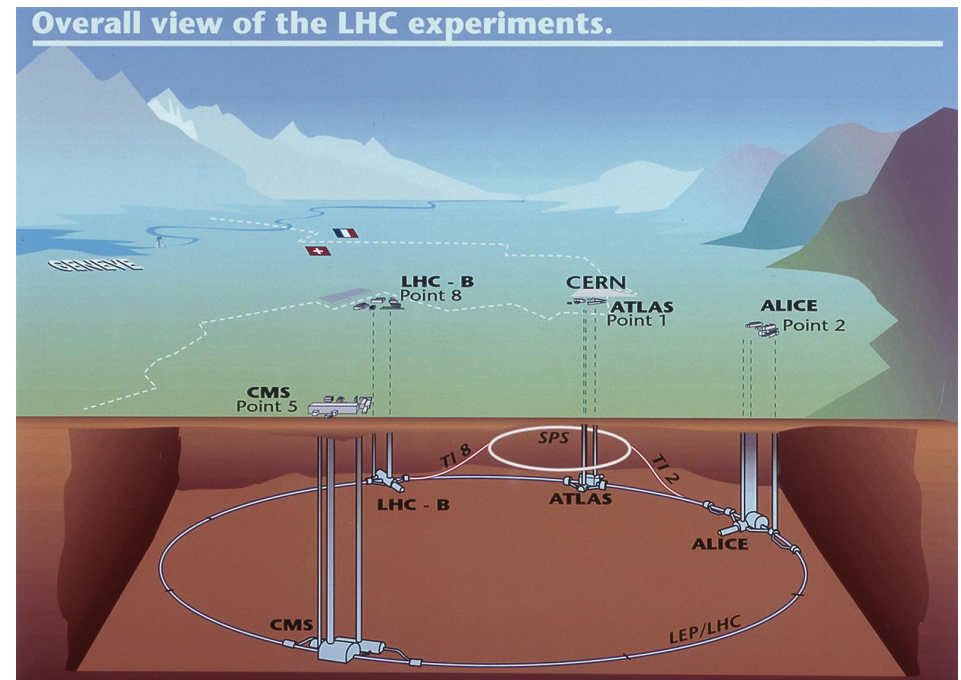
Tommaso Boccali  
INFN Pisa

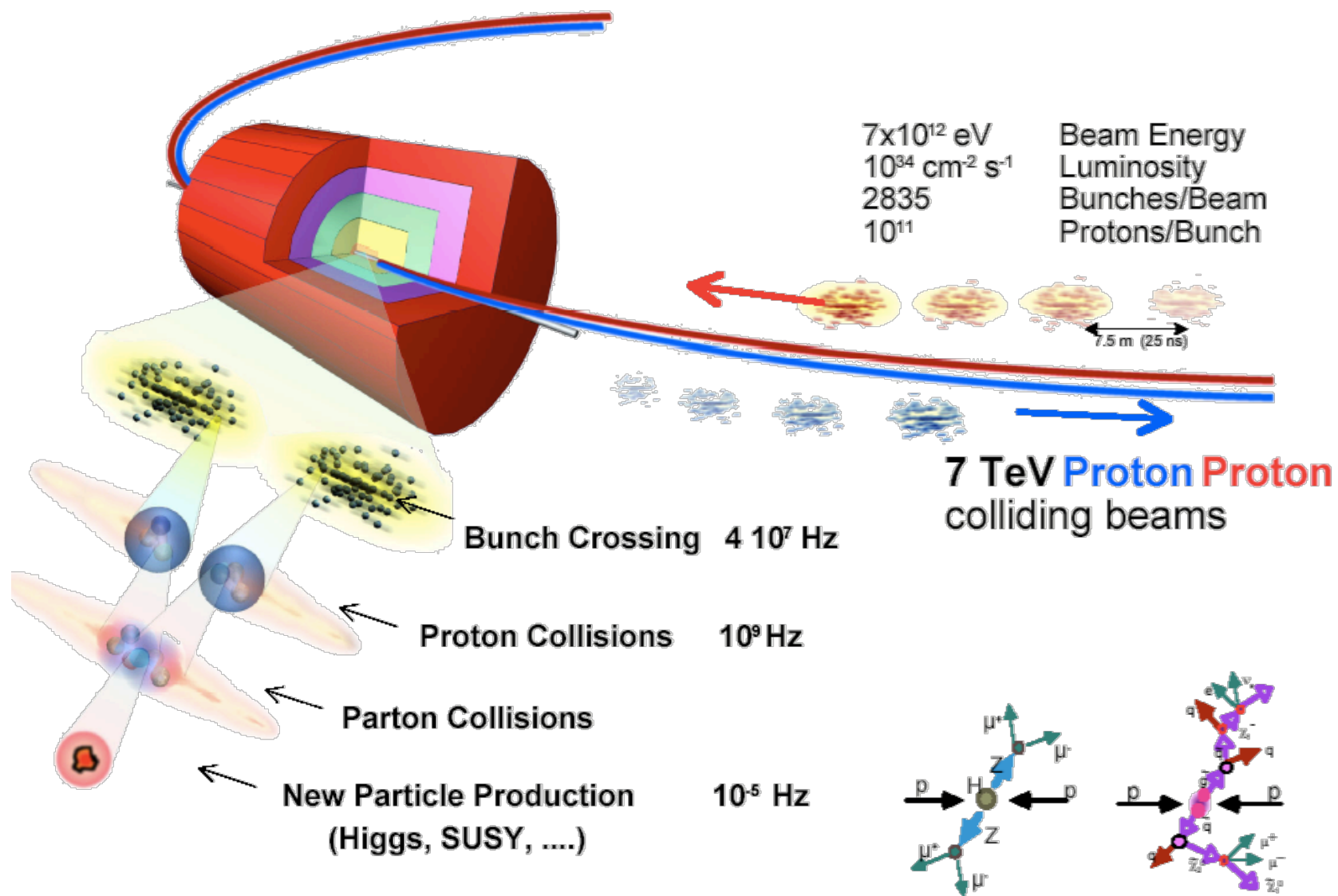
# Outline

1. High Energy Physics (HEP) experiments
  - Why do we realize them?
  - How are they built (with a focus on LHC)
  - Rough estimate of necessary resources
    - Some words on typical units of measurement
2. How to handle lots (lots!) of data
  - Possible solutions
  - The GRID solution
3. Are current computing models ok?
4. How will they evolve?
  - 2015
    - 2020
      - 2025 (???)

# HEP experiments with a computing eye

- Let's focus on LHC experiments
  - Simply, they are the latest entered into a production state, and thus the most advanced





**Selection of 1 event in 10,000,000,000,000**



# LHC Collider: basic parameters

Some of the more important parameters.

- Particles used: Protons (in proton- proton collisions) and heavy ions (Lead 82+)
- Circumference: 26659 m.
- Injected beam energy: 450 GeV (protons)
- Nominal beam energy in physics: 7 TeV (protons)
- Magnetic field at 7 TeV: 8.33 Tesla
- Operating temperature: 1.9 K
- Number of magnets: ~9300
- Number of main dipoles: 1232
- Number of quadrupoles: 858
- Number of correcting magnets: 6208
- Number of RF cavities: 8 per beam; Field strength at top energy = 5 MV/m
- RF frequency: 400 MHz
- Revolution frequency: 11.2455 kHz.
- Power consumption: ~180 MW
- Gradient of the tunnel: 1.4%
- Difference between highest and lowest points: 122 m.

LHC can accelerate protons and ions (independently in both rings)

Circumference: 27 km

Nominal beam energy at collision point: 7 TeV

Moreover

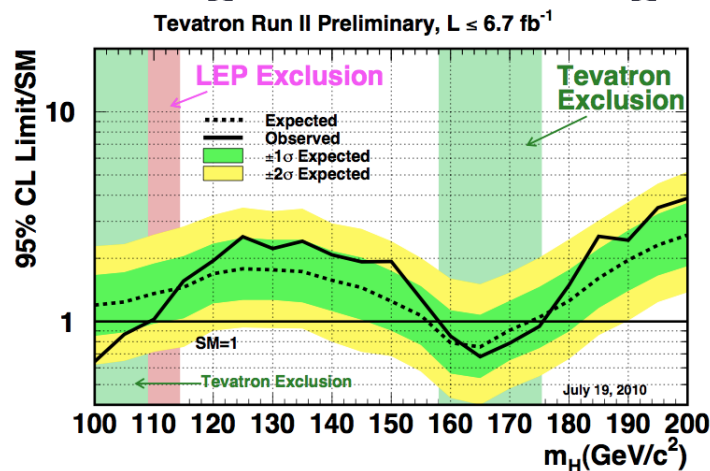
- Up to ~2800 bunches at the same time per ring
- Up to  $10^{11}$  protons per bunch
- Collisions every 25 ns

# Why do we need such extreme parameters?

- I will try to show you this is not “**since it is cool**”, but in order to reach the physics result
- LHC was built having in mind a very rich physics program, but with a clear focus on two possible fields
  - Higgs’ boson discovery & physics
  - Search for physics beyond the Standard Model
- Both fields are by no means “new”, and has already been attempted at least in the last two “discovery machines”: LEP and Tevatron
- So we knew in advance where that physics was NOT to be found, and LHC was thought and built mostly in order to explore the same physics in new energy regions.

# Let's just focus on Higgs Boson: where to search for it

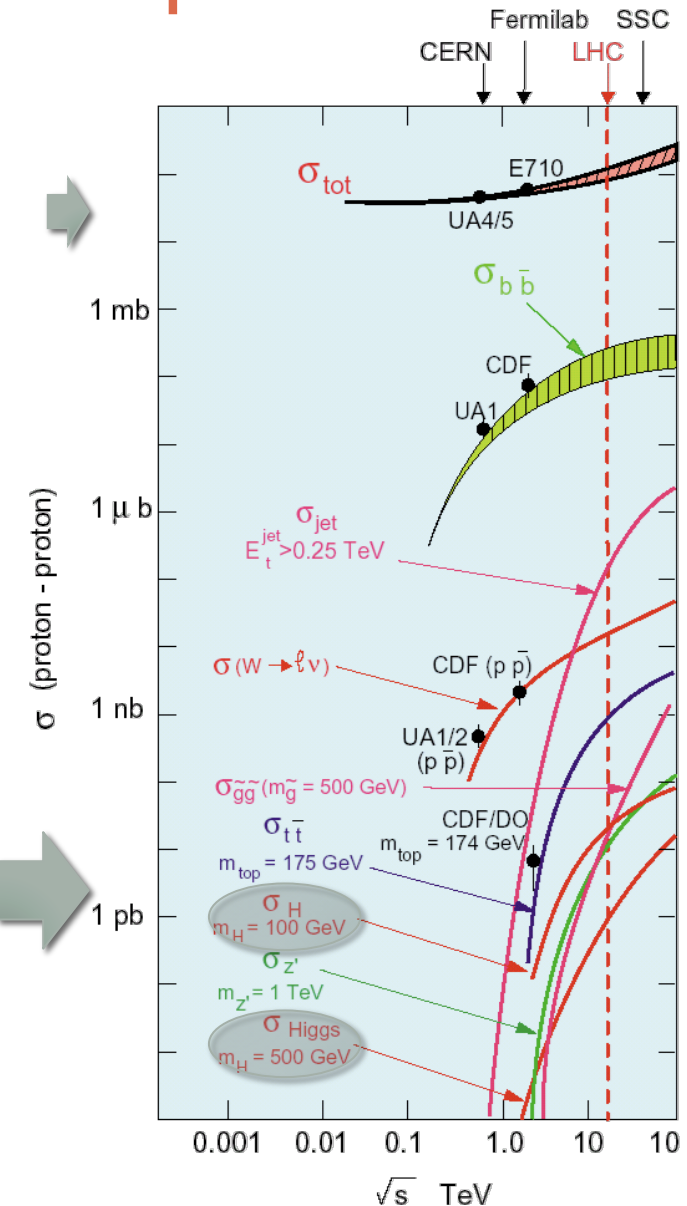
- After LEP and Tevatron, we knew quite well where NOT so search for it
  - LEP: lower mass limit  $\sim 115$  GeV (direct exclusion)
  - LEP: most probably below 200 GeV (indirect limits, depending on many theoretical assumptions)
  - Tevatron: + not in the range between  $\sim 160$  and  $\sim 175$
- Strong theoretical arguments against an Higgs boson



The nice feature of standard Higgs searches is that once you have (postulate) the mass, all the other parameters like couplings, production, decay rates are known (its mass is the last unknown parameter in the standard model), hence one can plan on Higgs characteristics

# Higgs boson production: to the problem's root

- Higgs production cross section (how probable to create one) increases very sharply with collider energy
- As you should know, the actual number of produced events in a given process is proportional to its **cross section**, and the collider **luminosity**
- **$N = \sigma \times L_{\text{int}}$**
- Where  $L_{\text{int}}$  is the integrated luminosity an experiment has been given
- Quite varying with the mass, but the typical Higgs production cross section is  $\sim 1$  pb @ a 14 TeV collider
  - @ 1 TeV collider it would be  $\sim 100$ -1000 times lower, this is the reason why a direct positive discovery at Tevatron was basically hopeless



# And then, which collider parameters do we need?

- In turn, integrated luminosity is the time integral of the instantaneous luminosity
- $L_{\text{int}} = L_{\text{inst\_average}} \times (\text{data taking seconds})$
- And again,  $L_{\text{inst}}$  is

$$L = \frac{f \sum_{i=1}^{k_b} N_{1i} N_{2i}}{4\pi\sigma_x^* \sigma_y^*}$$

$f$  = revolution frequency (c/27 km)

$N_{1i}, N_{2i}$  = number of protons in  $i$ -th bunch

$k_b$  = number of bunches

$\sigma_x, \sigma_y$  = transversal dimension of bunches in the colliding area

LHC
3564 (2808) bunches
$10^{11}$ p/bunch
$\Delta t = 25$ ns
$\sigma_{x/y}^* = 375.2(16.7) \mu\text{m}$ [IP1]



# Putting all together ...

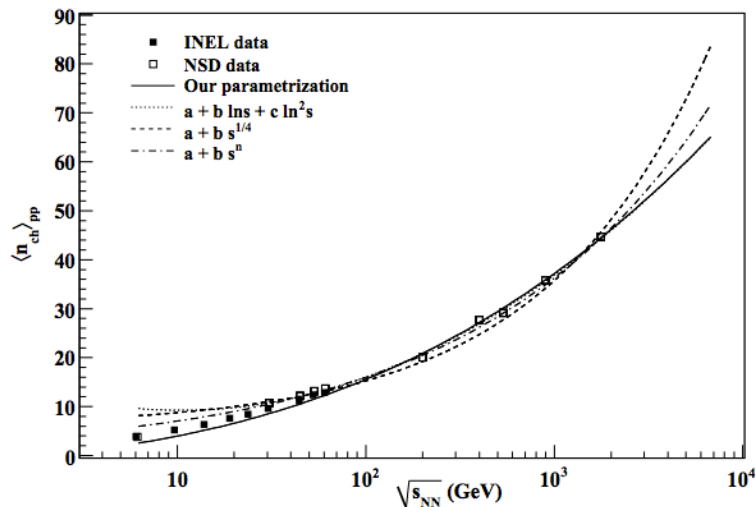
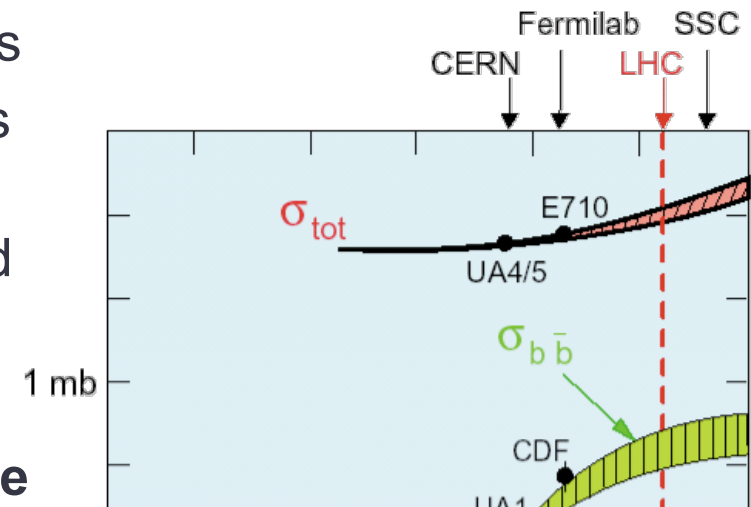
- If your goal is to have 100.000 Higgs in 5 years (per experiment)
- $L_{\text{int}} = 100 \text{ fb}^{-1}$  and then, scaling to the instantaneous lumi (assuming an efficiency factor  $\sim 5$  for shutdown periods, vacations, repairs, etc)
- $L_{\text{inst\_max}} = 100 \text{ fb}^{-1}$
- If you remember that  $1 \text{ b} = 10^{-24} \text{ cm}^2$

$$L_{\text{inst}} = 5 \frac{L_{\text{int}}}{15 \times 10^7 \text{ s}} = 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$$

- This is exactly what you get in the previous page formula wit LHC parameters
- **SO: the extreme LHC parameters are the only way to “guarantee” LHC would have been able to discover / exclude the Higgs boson in the energy range where we were searching for him.**
- **Any machine with lower parameters could have not been able to close the issue on the Higgs (if you want, not well spent money)**

# Executive summary on LHC

- It collides bunches of  $1e^{11}$  protons every 25 ns
- At each beams' collision,  $\sim 25$  hadronic events are generated
- Total = 1 billion hadronic collisions per second
- Each collision  $\sim 50$  primary particles on average
- **50 billion primary particles per seconds are generated into each experiment**



$100 \text{ mb} * 1e34 \text{ cm}^{-2}\text{s}^{-1} = 10^9 / \text{s} = 1$   
 hadronic event per ns = 25 hadronic  
 events per second





CMS Experiment at LHC, CERN  
Data recorded: Mon May 28 01:16:20 2012 CEST  
Run/Event: 195099 / 35438126  
Lumi section: 65  
Orbit/Crossing: 16992111 / 2295

Many interactions per crossing  
A huge Challenge for  
reconstruction, object ID and  
measurements

*Raw  $\Sigma E_T \sim 2 \text{ TeV}$   
14 jets with  $E_T > 40 \text{ GeV}$   
Estimated  $PU \sim 50$*



# An harsh environment ...

- So next step is: you need to be able and build detectors (“experiments”) able to sustain and use such a particle rate
- The same detectors have to survive (in 5 years)  $10^{18}$  primary particles, while being able to **identify/select** the 100000 Higgs which are produced, among  $3 \times 10^{16}$  collision events
- Selection factor =  $100000 / 3 \times 10^{16} = 1$  **“interesting” events every 300 billion interactions**

# LHC Experiments (the major ones)

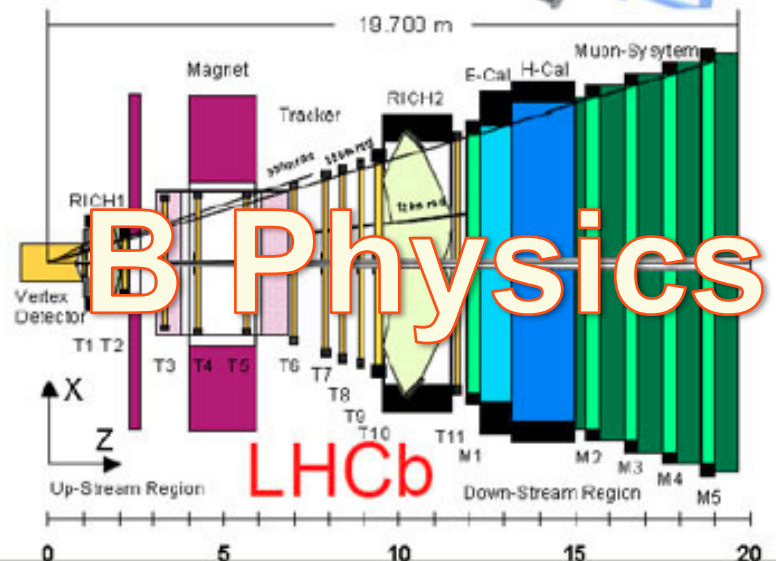
ATLAS

CMS

General Purpose



Heavy Ions



B Physics



# Detectors

- We have no time to describe here LHC detectors, and it is not even the scope of this seminar, but
  - The extreme event selection capability requires a strong precision on basic physics quantity measurements (like momentum, energy, position) for all the particles produced in the collisions
  - The only way we know to achieve this is via complex detectors, with many measuring channels (“acquisition channels”)
- Without distinguishing between the experiments, the average number of DISTINCT acquisition channels (“wires” going into a computer) is about 100 Million
  - And we can suppose each of these will produce 1 Byte per reading (naïve but not too unrealistic)

# CMS

## Superconducting magnet

Weight: 12,500 t  
Diameter: 15 m  
Length: 21.6 m  
Mag Field: 3.8 Tesla

## Calorimeters

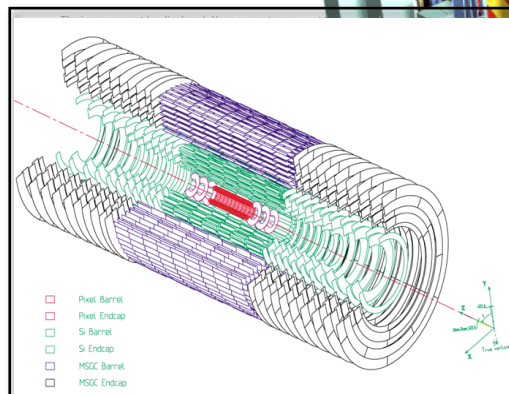
**ECAL** Scintillating  $\text{PbWO}_4$  Crystals 76000

**HCAL** Plastic scintillator

copper sandwich  
~1400

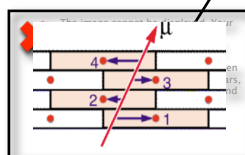
Return yoke for the magnet

## Tracker

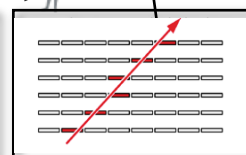


Silicon Microstrips  $\sim 10^7$   
Pixels  $\sim 6 \times 10^7$

## Mu chambers, BARREL

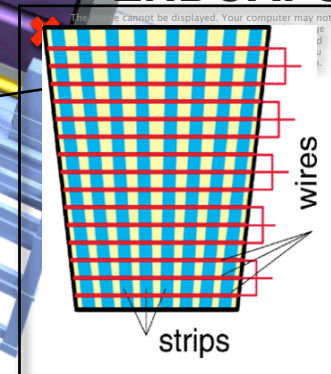


Drift Tube Chambers (DT)

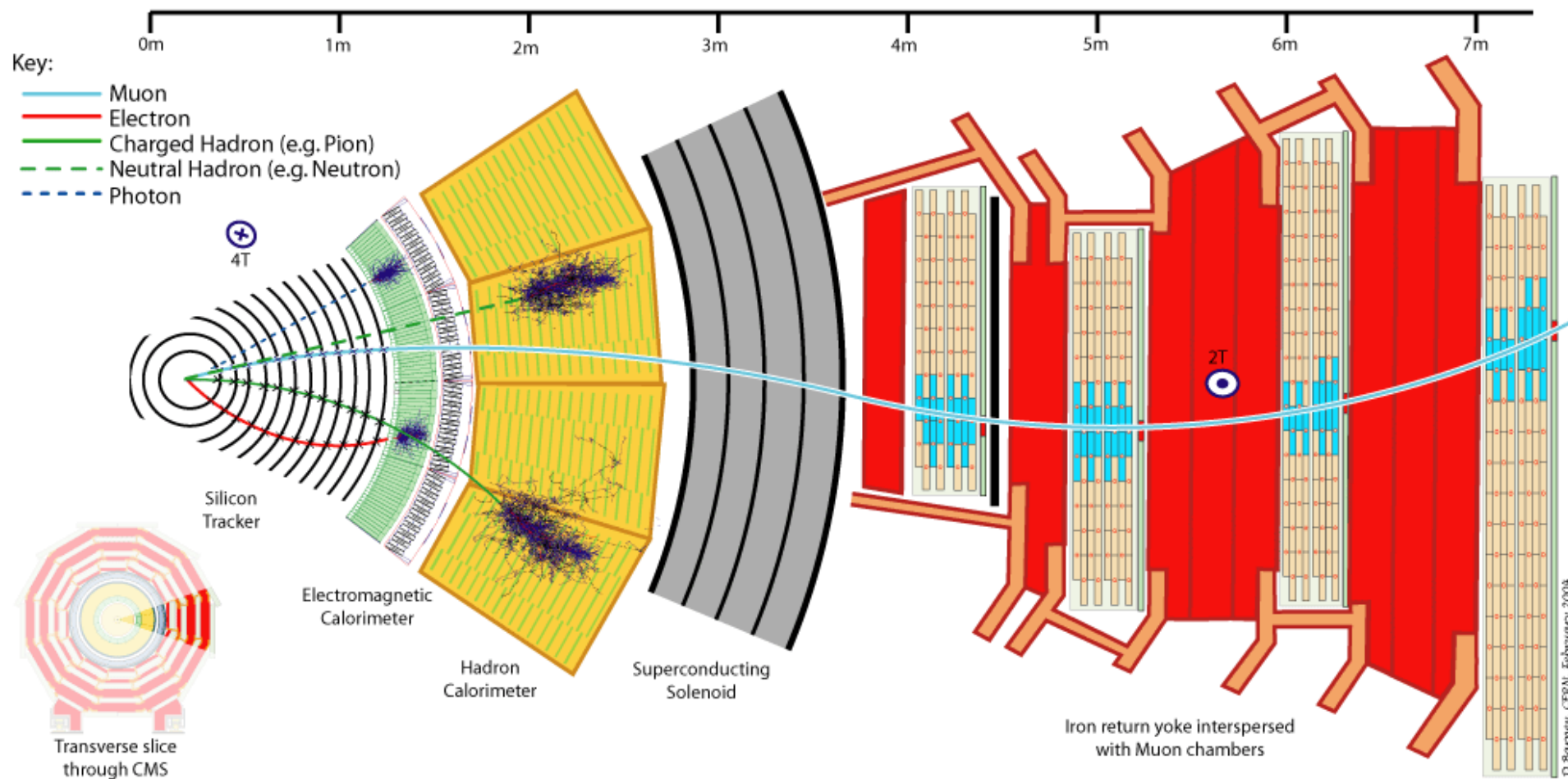


Resistive Plate Chambers (RPC)

## Mu chambers ENDCAPS



Cathode Strip Chambers (CSC)  
Resistive Plate Chambers (RPC)

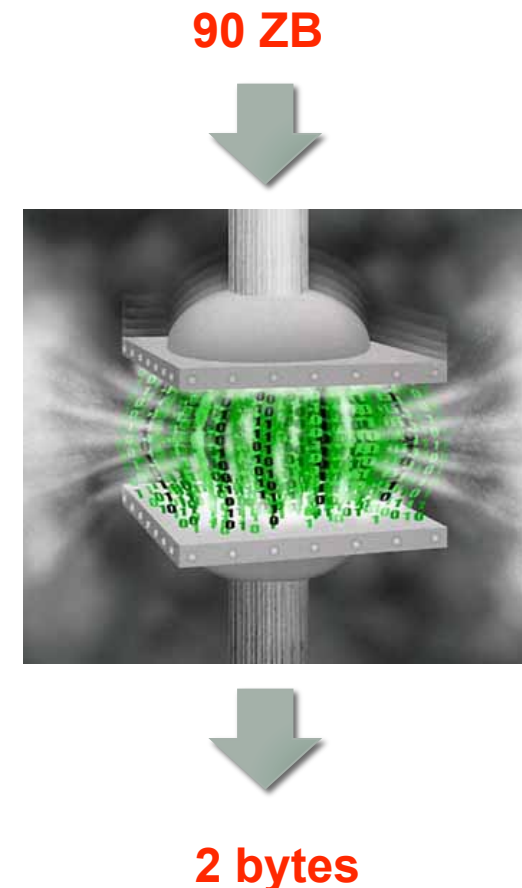


# Units of Measurements in HEP Computing

- Storage
  - 1 byte (B) = [0...255]
  - 1 GB =  $10^9$  B
  - 1 PB =  $10^{15}$  B
  - 1 EB =  $10^{18}$  B
- today = 1 HardDisk ~ 3 TB
- Network:
  - Gbit/s =  $2^{30}$  bit/s ~ 100 MB/s
- Today = National Research Networks (NREN) > 10 Gbit/s
- CPU:
  - 1 HepSpec06 (HS06) = unit specifically thought for HEP
  - today = 1 computing core  
core di esecuzione ~ 10 HS06
  - Today = 1 CPU (~8 cores)  
~< 100 HS06

# Which is the expected data rate?

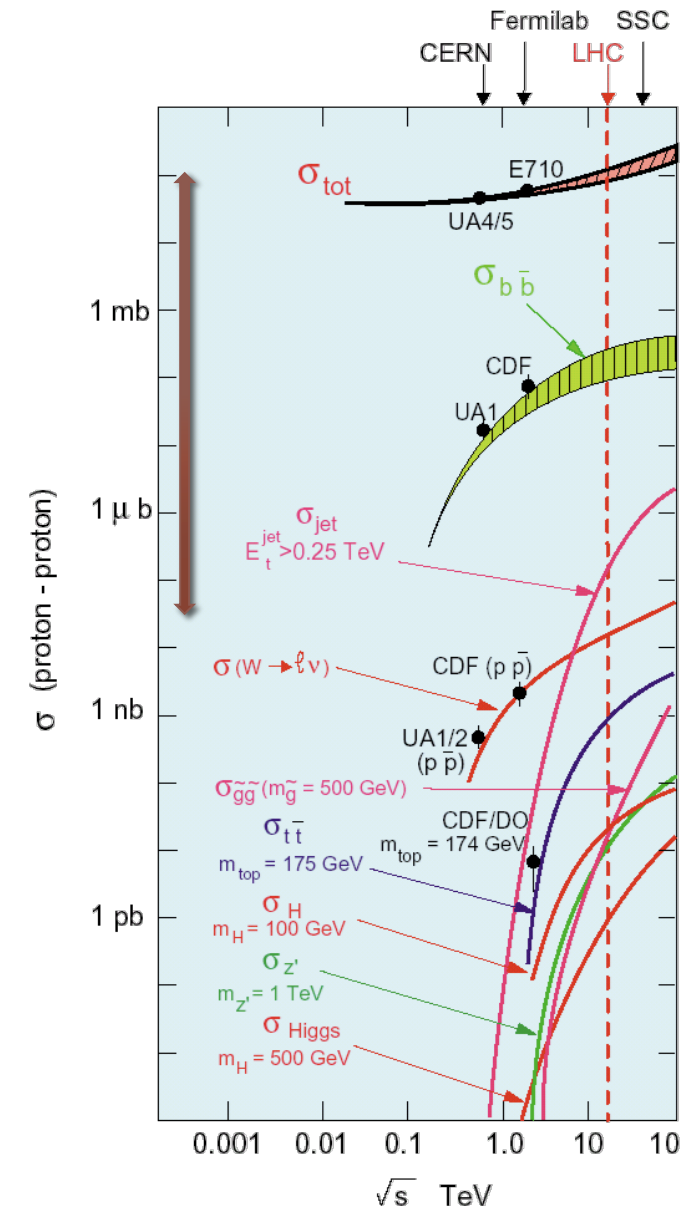
- 40 Million collisions per second \* 100 Million acquisition channels \* 1 Byte per channels per collision = **4 PB/s**
- Here we enter directly Computing Models realm: how to
  - Reduce 4 PB/s to something manageable
  - Analyze such a data flow and produce something human readable (a physics paper, for example)
    - Like: “Higgs Mass is 125 GeV”
  - Taking to the extreme, Computing Models are a means to reduce 90000 Exabyte to a couple of Bytes





# In reality ...

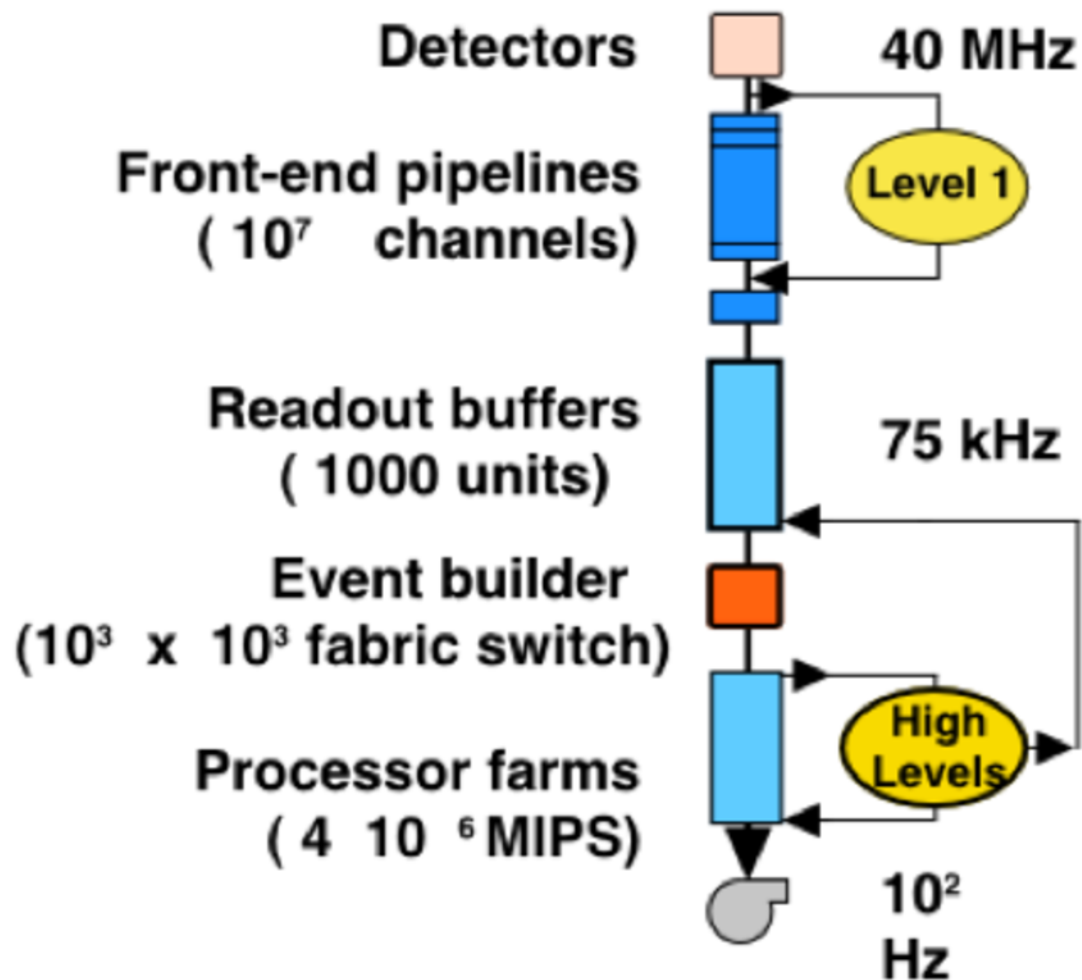
- It is absolutely clear no one will be able in the near future to handle 4 PB/s with IT systems, by some orders of magnitude
- It is also clear that the very bulk of this rate consists not so interesting events (like low energy QCD): **there are 5+ orders of magnitude between total cross section and interesting phenomena**
- The largest part of the events, if correctly identified, can be just thrown away
  - “if correctly identified”



# A match for the “trigger”

- While not strictly part of the Computing Model, the Trigger is essential to lower the data rate to something “manageable”, by identifying events NOT considered interesting before they hit any computers
  - Which is untrue since trigger also contains computers, but ...
- A trigger works by using a subset of the same signals we were considering, with a lower precision to increase selection speed which must be “real time”
- Many triggering strategies, with different details, can be used, but for what concerns us they are not too different from the one used by CMS in next slide.

# The trigger



- Input = 40 MHz (1/(25ns))
- Custom electronics select and reduce down to ~100 kHz (selection factor ~400)
- A second system, **based on commodity CPUs**, which works on semi-optimal quantities, goes down by another ~1000 to O(100 Hz)

# How good is a Trigger? Metrics:

1. Must be able to decrease the actual data rate from 4 PB/s to something manageable (today a few hundreds MB/s)
2. Must have decent efficiency on (like 10% or more) on events of physics interest
3. Must have high rejection (like  $\sim 10^5$ ) on not interesting events
4. Must work in real time or close (CMS = 30 ms at most)

Process	$\mathcal{E}_{L1}$	$\mathcal{E}_{HLT}$	$\mathcal{E}_R$ HLT
$Z/\gamma \rightarrow l^+l^-$	0.49	0.39	0.81
$t\bar{t}$	0.70	0.39	0.56
$W + jets$	0.57	0.42	0.72
$wtb$	0.61	0.36	0.59
$Zbb \rightarrow b\bar{b}\tau^+\tau^-$	0.44	0.19	0.43
signal $m_A = 200, \tan\beta = 20$	0.60	0.42	0.70
signal $m_A = 300, \tan\beta = 20$	0.78	0.63	0.81
signal $m_A = 400, \tan\beta = 20$	0.86	0.75	0.86

A Tau lepton trigger  
Typical efficiencies on  
selected channels

Table 4.4: Selection efficiency at L1 and HLT. The last column contains the HLT trigger efficiency relative to the L1 accepted events.  $\mathcal{E}_R = N_{HLT}/N_{L1}$  where  $N_{HLT}$  is the number of events passing the HLT and  $N_{L1}$  the number of events selected at Level-1.

# Decrease in data rate: not only trigger

- We said we work under the assumptions that each detector has ~ 100 Million acquisition channels, 1 Byte each per event
- Reading all of them is impossible, but also useless: most will not have values resulting from having been hit by a particle, but some form of **noise**
- **Zero Suppression** is the process with which on board detector electronics is able to detect null results (only due to noise), and transmit only real results
- Final event dimensions scale down by a factor 100 thanks do this for proton-proton collisions, 10 for Heavy Ions collisions
  - In what follows we will assume that event size is ~ 1 MB in pp, ~ 10 MB in HI



# Let's dive into Computing Models

- Fast summary of parameters
  - Rate of selected events:  $O(100)$  Hz
  - Typical dimension of each event:  $O(1)$  MB
  - Seconds of data taking per year:  $O(1e7)$  s
- Amounts to :
  - 1 Billion events per year
  - 1 PB per year of “RAW” data
- Much lower than the initial figures, manageable ....
- **Now what?**

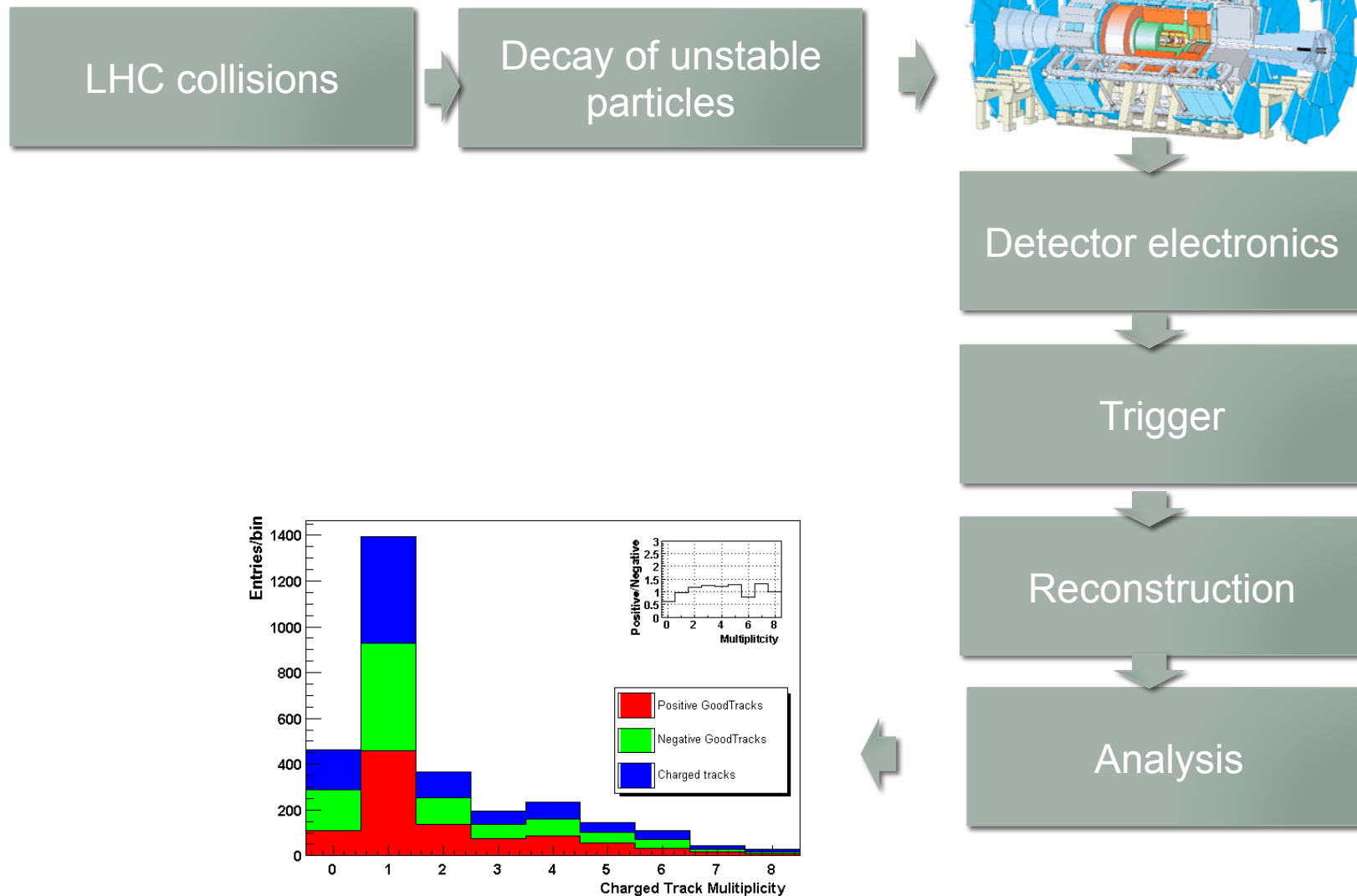
# Typical data workflow

- A physicist (apart from a few cases) is not able to interpret directly the RAW data from the detector
- He is used to think in terms of Particles, Jets, Decay chains, ..
- The process which allows for the interpretation of RAW data in terms of physical objects is called “reconstruction”, and it is usually CPU intensive.
- So: we do not have only the too-much-data problem, but also the too-much-cpu ...

# But before ... Simulations

- Up to now we spoke just about Data from the experiments
- In reality, this is not all of it. HEP dynamics, while in theory quite well known, in practice does not provide an analytical solution from the initial high energy collision to hadronization, decays, and finally stable particles.
- The only viable method is to generate statistically distributions via a **Monte Carlo method**, and compare these with the data
- In practice: events are “generated” sampling theoretical models with high statistics, and the events are then formatted to look as close as possible identical to the data events. **In this way, a 1-to-1 comparison can be cast between data and simulated events**

# Reality



# Simulation

Theoretical model

Simulation of decays  
of unstable particles

Simulation of  
interactions particle-  
detector

Geant  
4,...

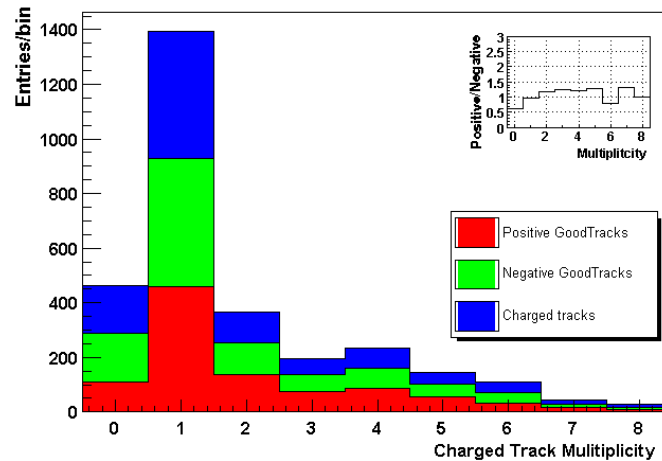
Pythia,  
...

Simulation of  
detector electronics

Trigger

Recontruction

Analysis



# Simulations

- As a consequence, theoretical estimates are not given to the experimental physicist as equations or such, but as simulated events which
  - As number
  - As size
- Are as close as possible to real data
- An accurate description of the models (due to its sampling) requires that the number of simulated events cannot be too small; they are typically in number as high as the real data events, if not more.
- Storage and CPU needs to store and analyze simulated events is not smaller than the one for data
  - Our approximation: we need to scale by at least 2x all the computing figures we have given up to now



# CPU needs in HEP

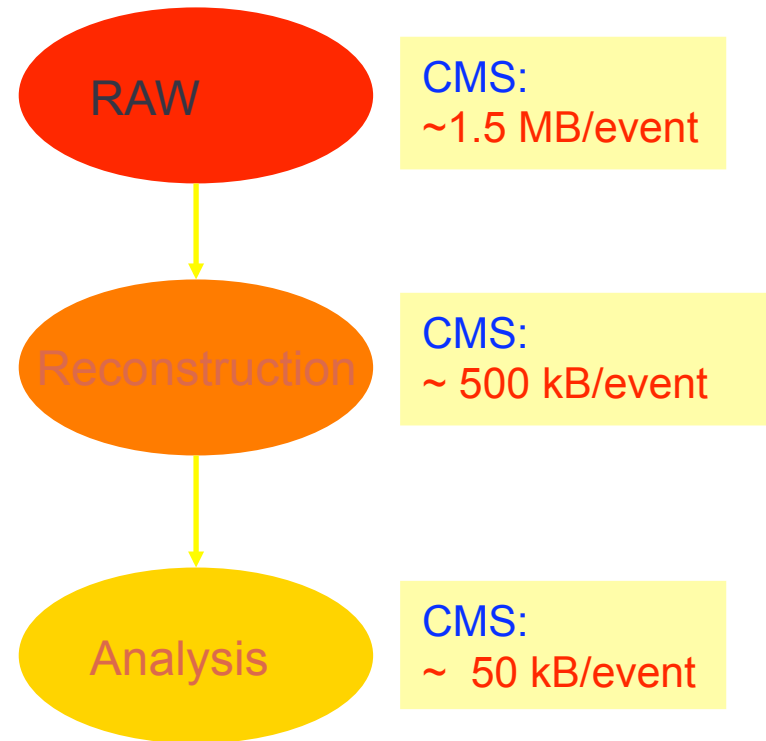
- The most important use cases are
  - Event reconstruction: its CPU need varies per experiment, but a reasonable estimate is 10 sec/event on today's CPU
    - 100 sec x HS06/ev
  - Event simulations: simulation of interaction of particles with matter (Geant4, mostly)
    - 500 sec x HS06/ev
  - Final data analysis (fits, final selections, result extraction, etc etc )
    - 1-10 sec x HS06/ev
- Summing all together:

# Official experiment figures

Experiment	Size RAW (MB)	Size RECO (MB)	Reduced size (analysis)	Reconstruction (sec.HS06/ev)	Simulation (sec.HS06/ev)	Analysis (sec.HS06/ev)
ALICE	1	0.04	0.004	25	150	2-64 <b>pp</b>
ALICE	12	2.5	0.25	3000	70000	30-1000 <b>HI</b>
ATLAS	1.6	.5	.1	60	400	2
CMS	1.5	0.5	0.05	100	180	1
LHCb	0.025	0.075	0.025	10		1

# Data Tiers: RAW, RECO, Analysis

- Experiments plan to implement a hierarchy of Data Tiers
  - **Raw Data:** as from the Detector
  - **Reconstruction:** contains reconstructed “Physics” objects(jets, tracks...)
  - **Analysis:** a subset of reconstruction, sufficient for the **large majority of “standard” physics analyses**
    - Contains tracks, vertices etc and in general enough info to (for example) apply a different b-tagging
    - Can contain very partial hit level information



# So a single data taking year ....

- **Storage**

- **Data:**

- 1 PB RAW (x2 for a backup copy)
    - 0.5 PB reconstructed

- **MonteCarlo**

- 1 PB RAW
    - 0.5 PB reconstructed

- ~5 PB/year

- **CPU**

- **Data:**

- $1e9 \text{ ev} \cdot 100 \text{ sec} \cdot \text{HS06}/\text{ev} = 1e11 \text{ sec} \cdot \text{HS06} = 10000 \text{ HS06}$  for the entire year
    - (plus x3, since data re-reconstruction happens frequently, as soon better calibration data becomes available)

- **MC**

- 10000 HS06 (x3) reconstruction
  - 50000 HS06 simulation

- **Analisi (MC + DT):**

- $1e9 \text{ ev} \cdot 2 \cdot 10 \text{ sec} \cdot \text{HS06}/\text{sec} \cdot N = 2e10 \text{ sec} \cdot \text{HS06} \cdot N$
  - Where N is the number of independent analyses, can be very high (~100)

- **TOTAL:  $3e11 + 3e11 + 5e11 + 5e11 = \sim 2e12 \text{ sec} \times \text{HS06}$**

- **Today they are**

- 2000 HDD
  - $1e5 \text{ HS06} = 10000 \text{ computing cores}$

- **Three years ago they were at least 3x higher**

- **.. And these are per experiment!**

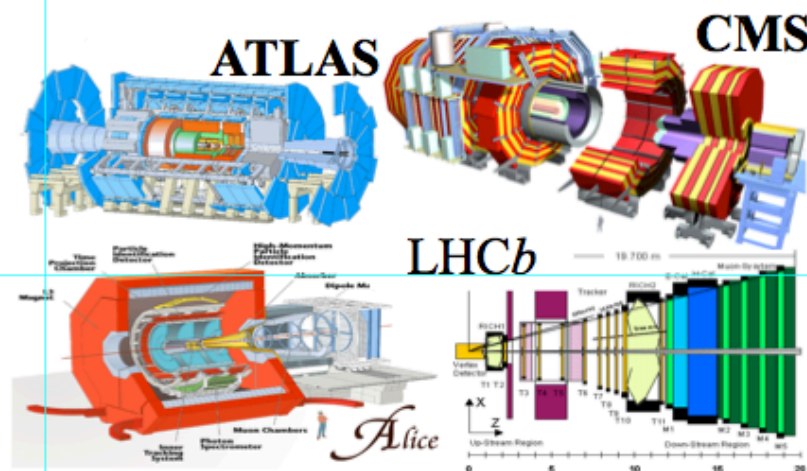


## Ultime stime ... (LHCC)



### ■ Per i 4 esperimenti, in totale:

- **CPU ~ 100MSI2k**
  - ~ 100000 CPU attuali
  - ~ 25000 CPU 2007
- **Disco ~ 40 PB**
  - 100.000 HD da 400 GB
- **Nastro ~ 40 PB**



### ■ Quali soluzioni sono possibili per il calcolo a LHC?



# After many estimates, which is the situation today?

Experiment	CPU (kHS06)	Disk (PB)	Tape (PB)
<b>ALICE</b>	391	37	44
<b>ATLAS</b>	780	93	63
<b>CMS</b>	636	59	76
<b>LHCB</b>	190	13	17

Resources used by experiments in 2013

Factor ~2 everywhere, mostly due to the fact that LHC was somehow more successful than expected



# We indeed are much bigger than anything before!



## Il Problema



- Dimensione del problema mai affrontata prima, sia dal punto di vista della necessita' di storage che di calcolo

	<b>ALEPH 1995</b> 	<b>CDF 2004</b> 	<b>CMS 2007</b> 
<b>Dimensione dei dati raccolti</b>	<b>1 TB = 1000 GB</b>	<b>1 PB = 1000 TB x1000</b>	<b>~10 PB x10</b>
<b>Capacita' di calcolo (SI2k)</b>	<b>&lt;&lt;100k</b>	<b>1.4 M x50</b>	<b>&gt;25 M x20</b>

Nota: 1 PC attuale ~ 1 SI2k

# How to build on paper a Computing model in ~ 1995?

- When LHC computing models started to be sketched, a typical computer had
  - ~ 10 GB HDD
  - ~ 0.1 HS06 single core CPU
- You can understand what **leap of faith in technology** is needed to think that in 10 years you will be able to handle resources which, in 1995, were of the same size of the entire world IT resource
- That said, how to handle this amount of resources?

# Possibilities

1. **A BIG data center**
2. **Many small data center**

# A big data center



- A large building with ~100000 computing cores, and 10000 HDD
  - Probably it would work; **Google** apparently has this kind of facilities
- But, the solution was considered not interesting, due to various reasons
  1. **A single point of failure** (if CERN goes offline, LHC computing follows...)
  2. **Political problems:** Member States were not so happy to finance “cash” computing at CERN (and in general, out of national boundaries)
  3. **Manpower:** difficult to find locally the large amount needed
  4. **(other) political problems:** member states wanted to increase their national expertise, not to finance Swiss ones ...



# Go distributed!

- During the '90s, as a pure IT concept, an alternative was born; the **GRID**
- In 5 minutes
  - Key concepts
  - Philosophy
  - implementations

# GRID – what are they?

## The Grid Vision (by Ian Foster)

“Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations”

- On-demand, ubiquitous access to computing, data, and services
- New capabilities constructed dynamically and transparently from distributed services

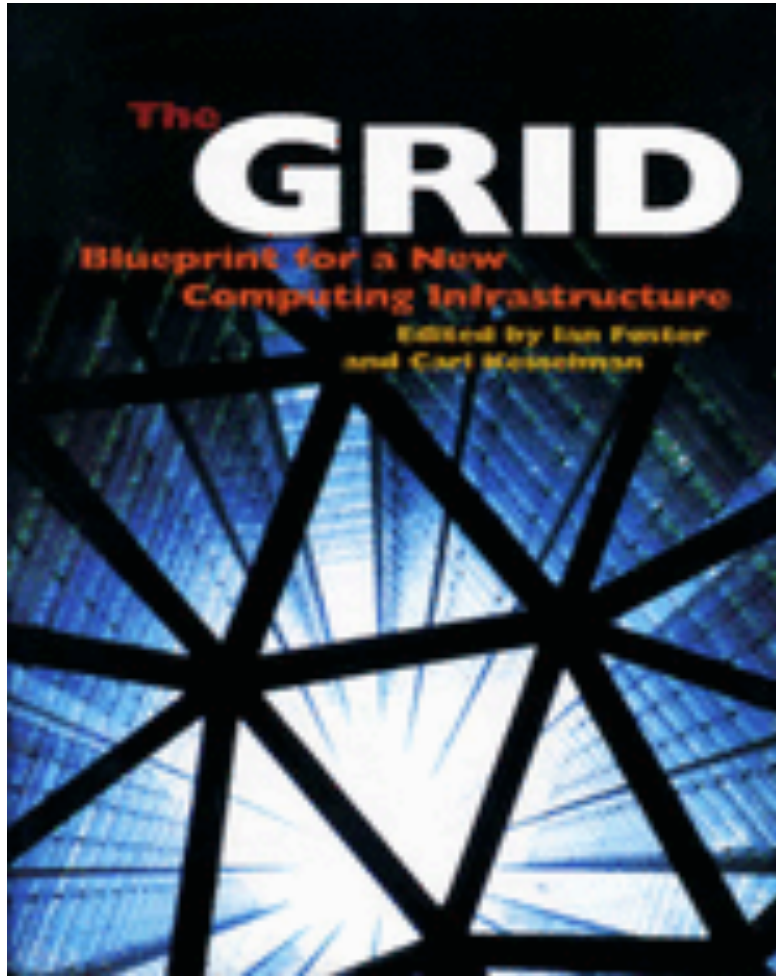
*“When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special purpose appliances”*

(George Gilder)

## More simply ...

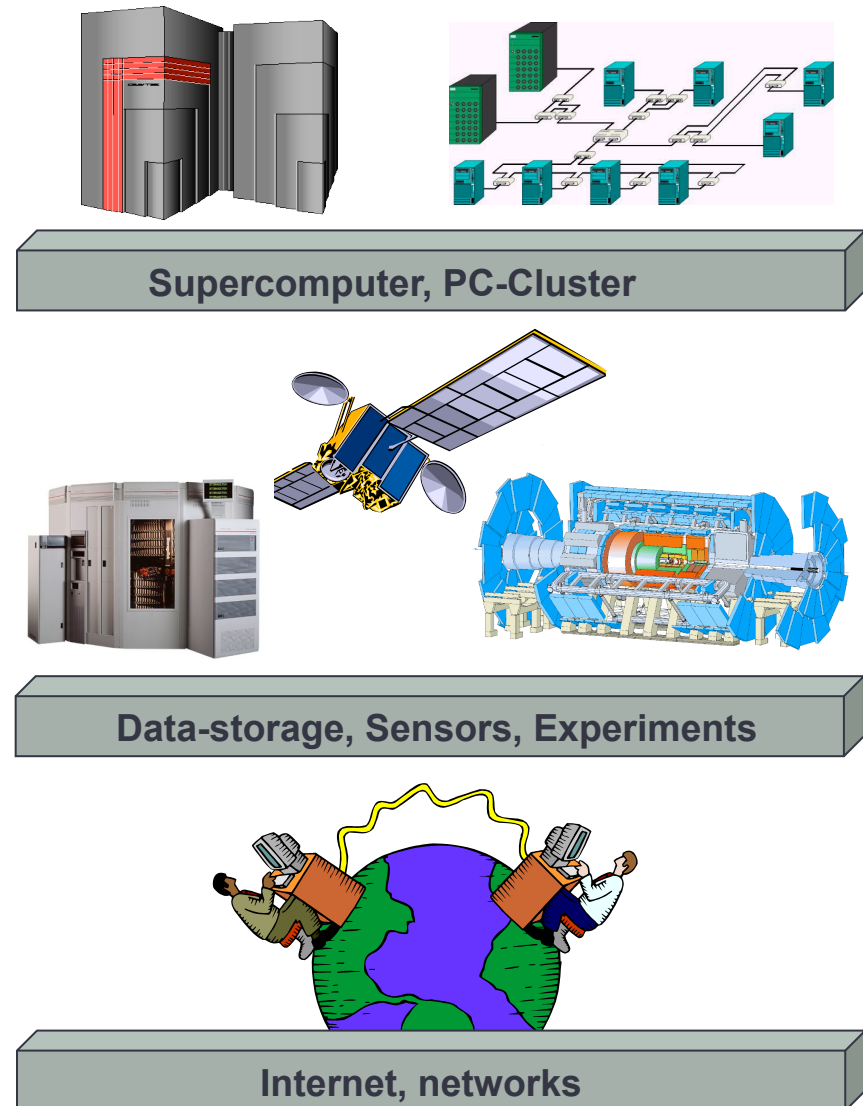
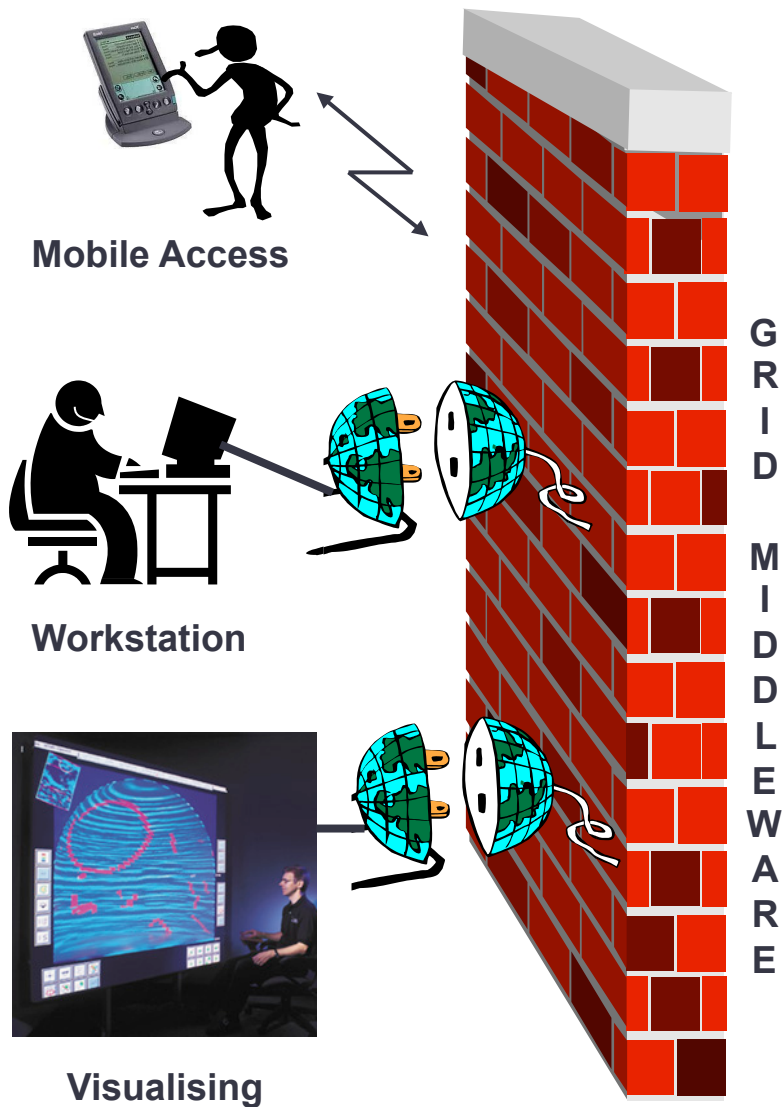
- **Give access to heterogeneous and geographically distributed computing, without being (too) aware of this**
- **GRID: they are named after the “power grid”**
- **For example:** Italy produces idro-electric and thermal power, moreover Italy buys power from outside (France, ...)
- But, when you need to use a blender, you do not need to care about
  - Which is the power source
  - Where was it produced
- You simply want and can access the power you have been given ( == you decided to pay)

# Formalization ...



1999:  
The GRID  
Blueprint for a new  
Computing Infrastructure

# The Grid metaphor





# In a nutshell ...

- Central Authorization/authentication
  - You will not need to request a “unix account” on all the machines in the world, but
    - You will need to be authenticated by a **Trusted Authority (Certification Authority)**
    - You will need to be authorized to the use of Experiment X resources by another service, which certifies the right to use them - **Virtual Organizations (VO)**
- GRID resources are seen from you as
  - **Storage Nodes** (Storage Elements, SE): they are accessible via standard authenticated protocols
  - **Computing Nodes** (Computing Elements, CE): basically single batch farms



■ ■ ■

- And, at least in some GRID implementations, some “resource brokering”
  - Given a computational task, find the “best place” where to execute it (on a planetary scale)
  - Given a filename, access it wherever it is (without realizing)
- **GRID ambition has been to have geographically distributed computing not different from local one, from a user point of view**

# GRID and LHC experiments

- So, distributed computing was chosen as the solution
- That given, how to organize LHC computing on it?
- **It turns out it is NOT as simple as divide the resources in 50 sites and use them (regardless the GRID)**
- There is a nasty aspect we did not cover for the moment: the **Network!**
- Again some rough HEP estimates, this time on the networking

# A single experiment networking needs

- RAW data =  $300 \text{ Hz} * 1.5 \text{ MB/s} = 450 \text{ MB/s}$
- Reconstructed data = at least **2x** (including reprocessing)
- MonteCarlo = as data, so factor **2x**
- Analysis = a rough estimate gives 1 Mbit/s/HS06, so **10 GB/s**
- Overall per experiment ~ **15 GB/s or O(150 Gbit/s)**
- In an ideal GRID environment, chaotically distributed among 50 sites

## ~2000: which networks were expected to exist?

- In many states it was before network deregulation: single actor, semi-monopoly
- Expected increase (also due to monopoly) less than a factor 2 per year, at a given price
- Pisa INFN as example: in 2000 it had a WAN connections via GARR (italian research network) topping at 8 Mbit/s. In the 5-6 years to the LHC start no way to get to 10 Gbit/s
- **Result:**
  - **It turns out it is possible to guarantee (== pay) only a small number of network connections, and require on these high performance**

# We need to be Data Driven!

- Even if GRID is used, **if we do so we are not really “location independent”**: and **not all the sites are equal** (since they are served with different connections)
- LHC Computing model becomes Data Driven
  - The activity a single site can carry on depends on the data it can access “locally”
    - A LAN activity, with no WAN consequences
  - Local data depends on its turn on how easy is to move data locally

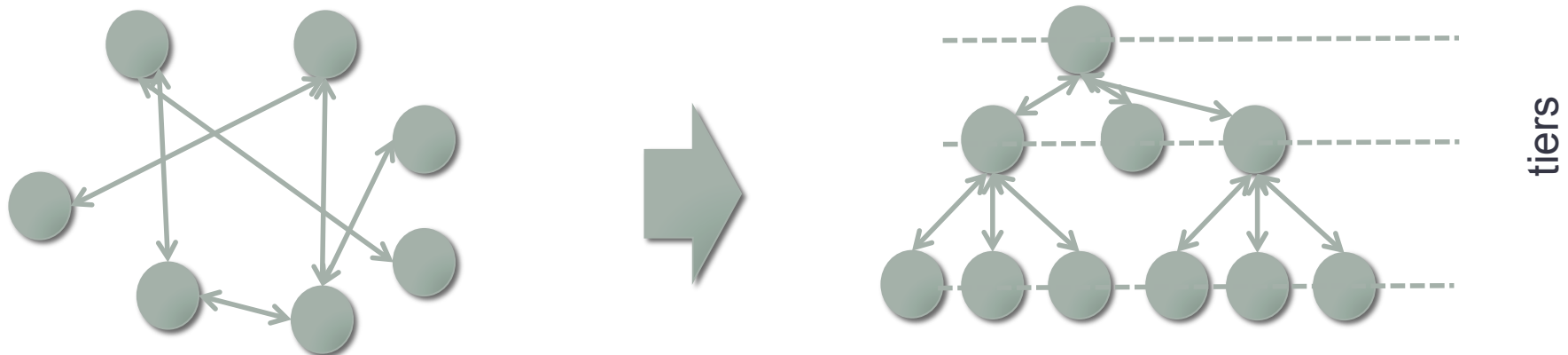
# An LHC Computing Model

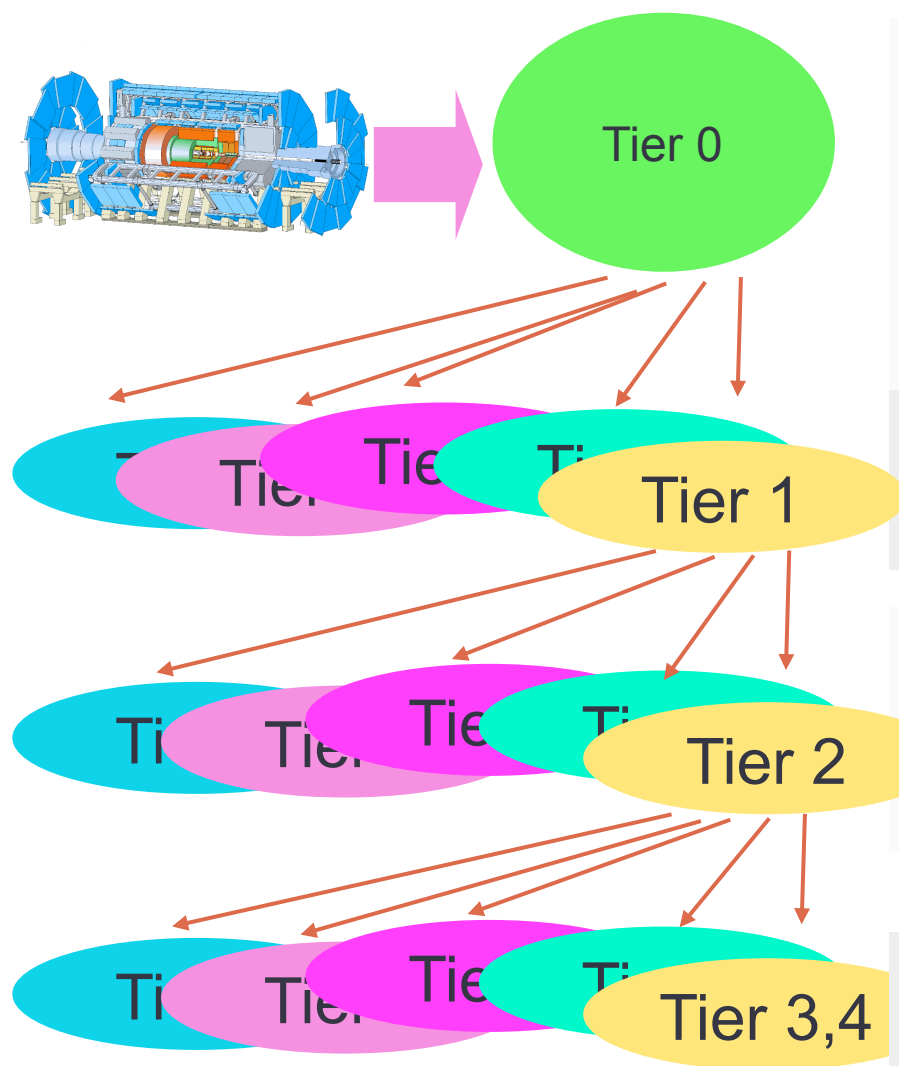
- A working group was formed (**MONARC**) to formalize a Distributed Computing Model for LHC
- Funding Ideas:
  - Have **predictable** network rates
  - Enforce “precious” data flows on **few links**
  - Define links you have to depend upon, in order to be able and **guarantee** them



# Outcome (early 2000s)...

- Distributed computing model, but in a **hierarchic structure**: hierarchy via “**computing tiers**”
- Hierarchic model: since (real) data originates at CERN, it must be have a central role. Data will flow from it to the other sites, in a pyramidal structure
  - **MC can in principle be generated in any place, but it will still need a central place for consolidation and traffic management**





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

# Other effects of being DataDriven

- Ideal GRID: if I need to process computational tasks (“jobs”), I will do it on sites where there are some not used CPUs. They will access data transparently via the network
  - This is BAD: this makes data paths not predictable. We cannot do it
- Hierarchical GRID model (“DataGRID”)
  - Jobs **just access local data** (local = already present in the same site/ cluster/ building)
  - ... but someone must have **preplaced** the data there!

# Some consequences...

- You need to prepare / put in operations a system to move data between sites. You cannot use on-demand since it is not predictable
- When moving data, it is absolutely necessary that the most precious ones (those which will require more access) are pre-placed
  - In the best sites (as for availability)
  - In the sites with more CPU
  - In more sites
  - When data changes from “interesting” to “not interesting” reverse actions must be taken
- Sites must be well balanced
  - A site with a lot of CPU and not enough disk will not be used efficiently
  - A site with few CPUs and a lot of storage will be overwhelmed with pending jobs

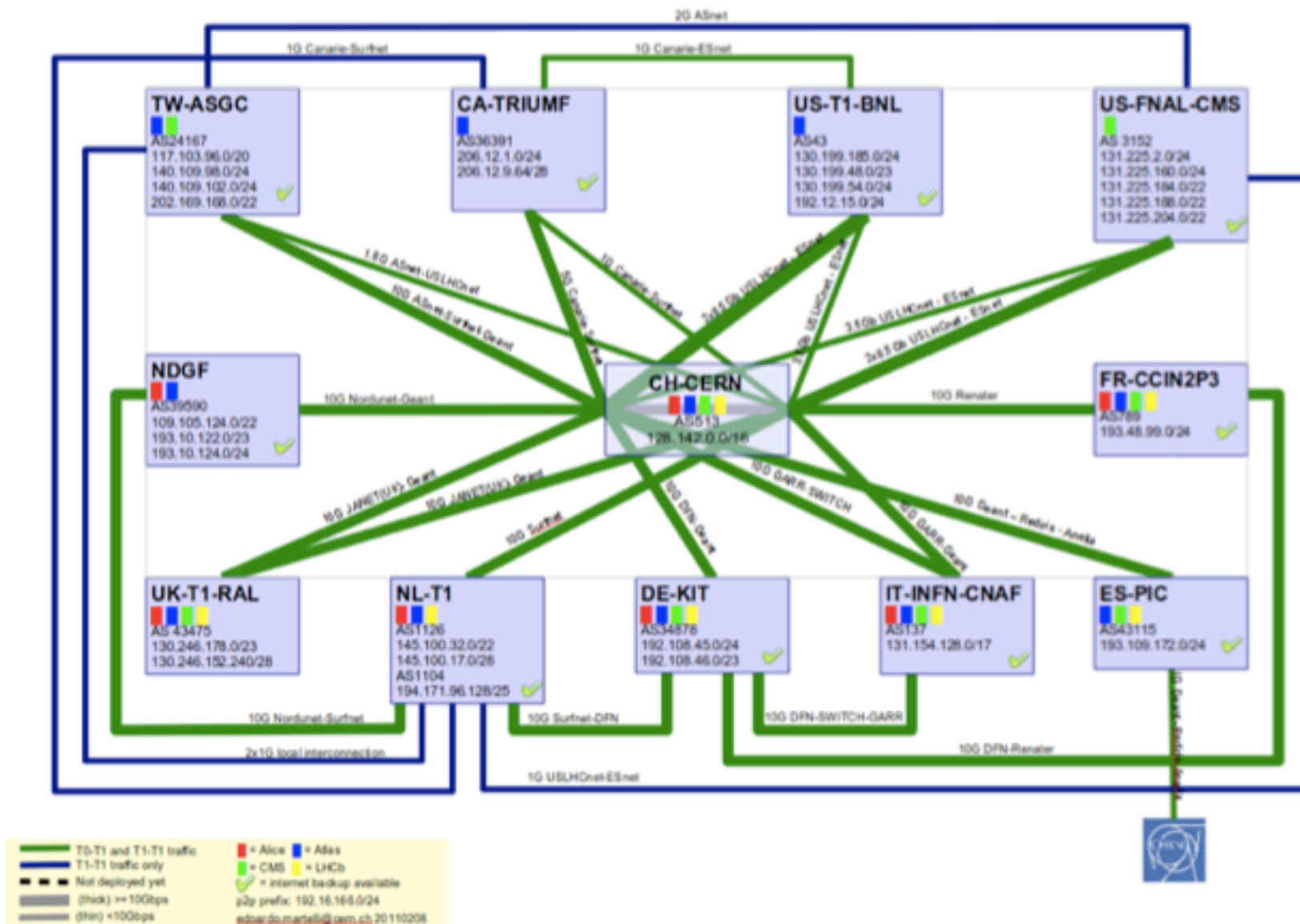
# Where a job should run ...

- The match-making between a site and a job is called “**Brokering**”, and in a **DataGRID** it is not a trivial activity
- When a job, which needs to access some data, is submitted, the possible sites able to process it are those with
  - **Free CPUs**
  - **With that data locally available**
- **On global scale you can do better: you have  $N$  jobs, which need to access  $M$  data files, on its turn available on  $M$  sites.**
- The global strategy which allows to use at best the sites is indeed called “Match-Making” strategy, and is a multi dimensional problem with no clear analytical solution

# Networks

- As we said, you can typically guarantee just a few links, those between Tier0 and Tier1s (which host the precious and not reproducible RAW data)
  - In a network language, this means private optical fibers should be used
  - ... and they cost a lot!
- LHCOPN: private operation network, guaranteeing at least 10 Gbit/s between CERN and the 10-12 Tier1s

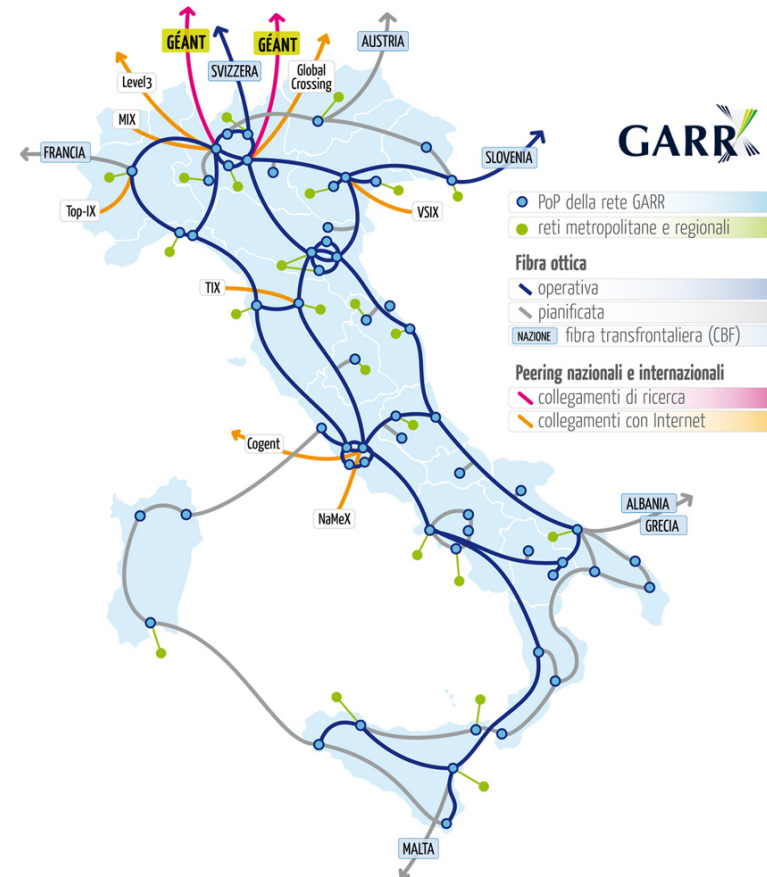
# LHCOPN





# What about T1-T1 and T1-T2?

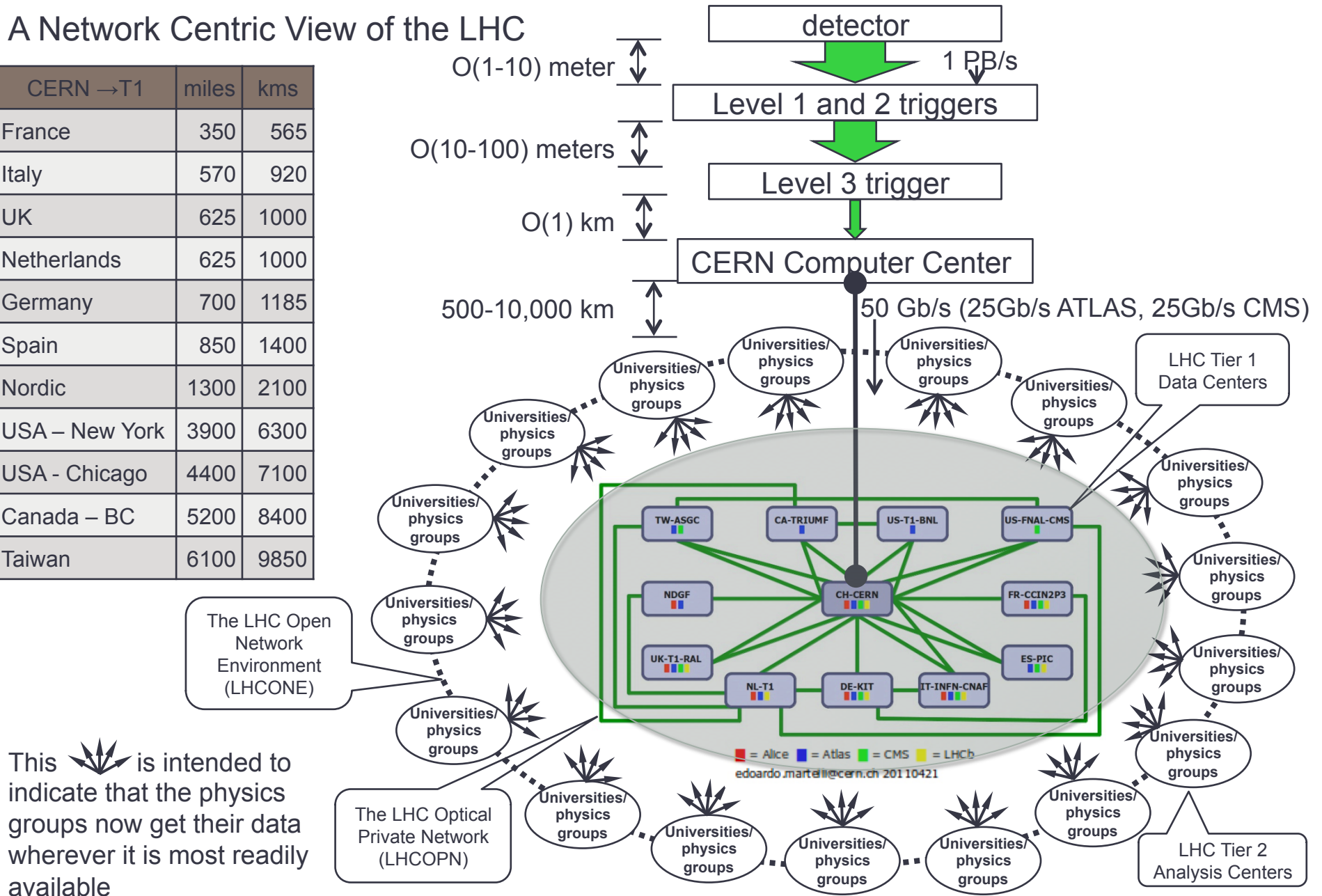
- Nothing guarantees, just based on what National Research Networks (NREN) were providing
  - no network provisioning: LHC traffic is just like any other research traffic
- For Example, in Italy our NREN is called GARR (Gruppo Armonizzazione Reti della Ricerca)
- The full LHC network topology is then:



## 6r

# 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



# So we have the Computing Model infrastructure

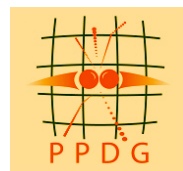
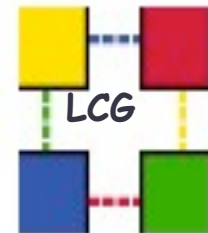
- We have GRID, we defined MONARC
- We have ~50x4 Computing Centres (the Sites)
- What defines a working system, which needs to have
  - Uniformity in the computing environment
  - Uniformity in the access protocols
  - Support for operations...
- We need a Worldwide coordination

# For example, GRID projects



- Are more than a few, in principle each with a different interface, Middleware ...

CrossGrid





# WLCG as the orchestrator

- “GRID” is a computing paradigm
- WLCG governs the interoperation since 2002 between the number of “concrete GRID implementations” (a number of, the main ones being OSG, LCG, NurduGrid, ...)
- WLCG was crucial in planning, deploying, and testing the infrastructure before 2010, and is crucial for operations now



## As of today, from REBUS

- CPU 1.8 MHS06 (~180k computing cores)
- DISK 175 PB (~80k HDDs)
- TAPE 170 PB (20-80k tapes)
- # Sites exceeding 200

ATLAS > 100k jobs/day  
CMS > 100 TB moved /day

Still increasing ...

# Executive Summary

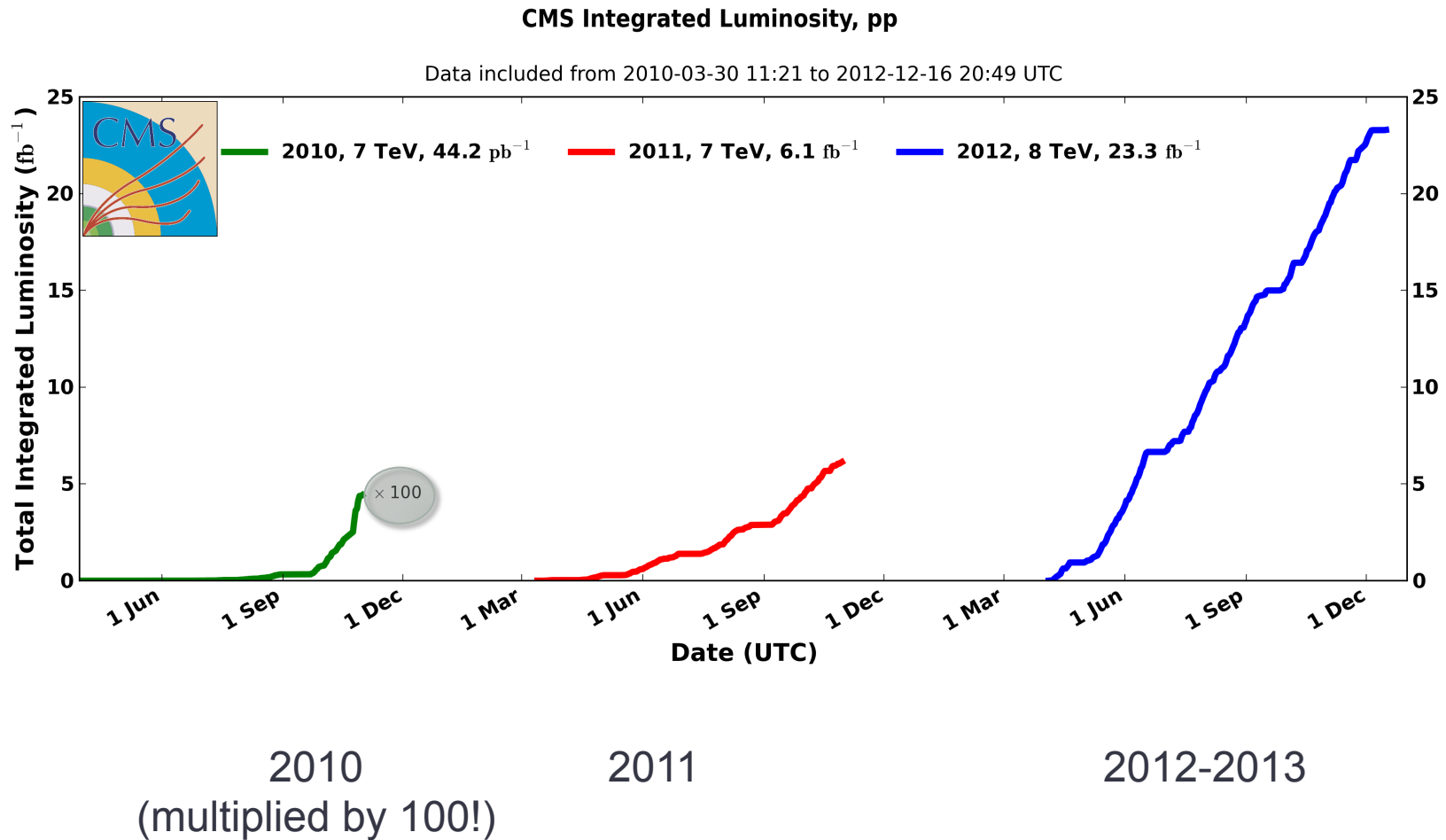
- We defined the amount of resources needed for LHC computing
- We decided where to deploy them, with which structure
- We have computational activities, and we defined where in the structure to perform them
- This needs organized data moving activities
- That is the 1995-2005 model, what happened in reality with LHC startup (2010)?

# Reality check

- 2005 became ... 2010
  - LHC magnet delivery was late
  - In 2008 there was the big magnet incident (~1.5 years for repair / recheck)
- Energy was reduced from 14 TeV to 7 and then 8 TeV
- These are the only real differences, all the rest was as expected or better
  - Better in this case means more data = more difficulties for computing



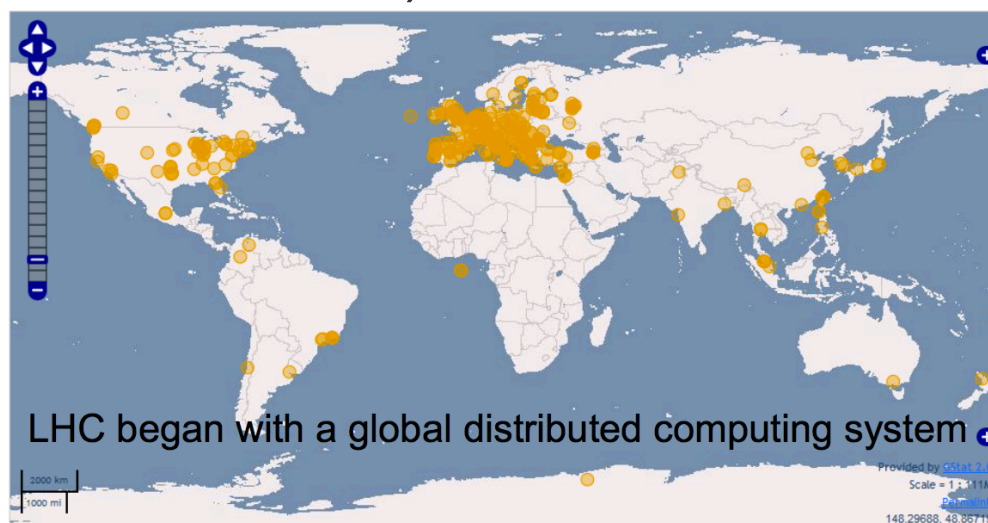
# LHC Run I



# Resources, events

- Real DATA: 1-5 Billion events per year per experiment
  - (should have been 1)
- Real Resources: (about 1.5-2x with respect to expectations)
- 200 Sites(1 T0, 13 T1, ~130 T2, ?? T3)

Experiment	CPU (kHS 06)	Disk (PB)	Tape (PB)
ALICE	391	37	44
ATLAS	780	93	63
CMS	636	59	76
LHCb	190	13	17



# Time to market!



- Handling LHC computing has surely been in these 3 years
  - Fatiguing (lots of manpower needed for services, support, data movement, job handling ...)
  - Complicated (the system has a huge number of degrees of freedom, it is hard to optimize)
  - Expensive (200+ sites)
- ... but it has lived up to Physicists' expectations
  - Jul 2010: first  $t\bar{t}$  events shown in Paris, 72 hours after having been collected
  - Jul 2012: "Higgs discovery day", with data shown collected up the previous week

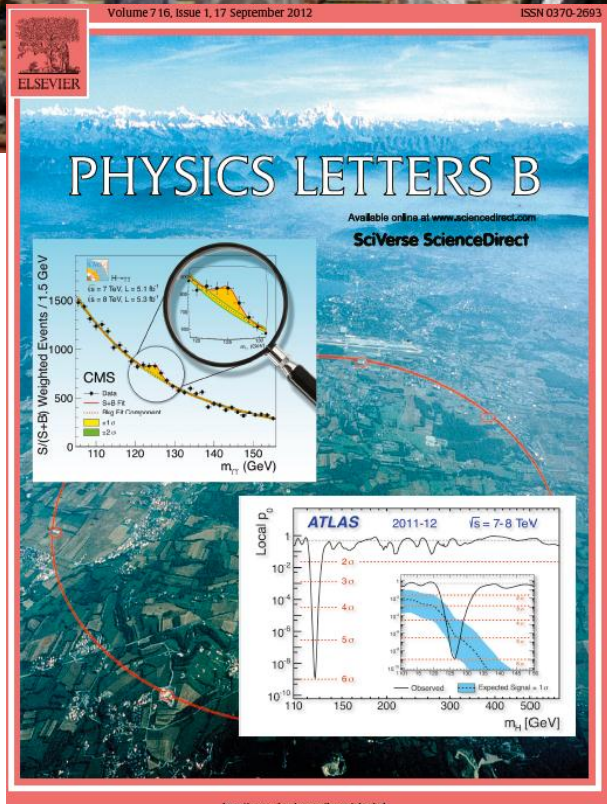
# Just a comparison

- CDF (Tevatron experiment): time needed for data to go from being collected to Analysis ~ some months
- Regardless the complexity, at least 10x faster at LHC
  - Due to very good online calibrations, but also to the computing model

# Publications

- They are the real metric for an experiment success
- Using CDS.CERN.CH (in three years)
  - **ATLAS = 257**
  - **ALICE = 63**
  - **CMS = 272**
  - **LHCB = 120**
- **By comparison, LEP experiments (in 12 years)**
  - **ALEPH = 321**

# Results...

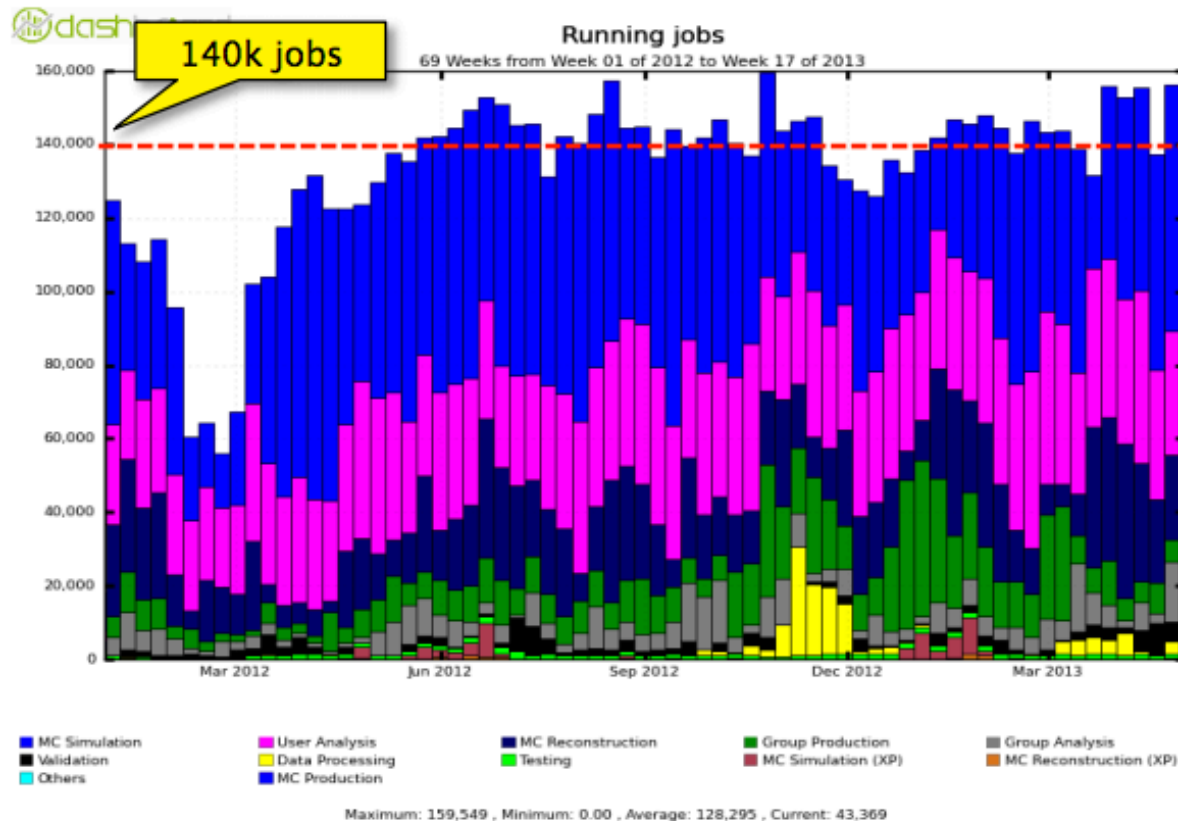


In summary

We have observed a new boson with a mass of  
 **$125.3 \pm 0.6$  GeV**  
at  
 **$4.9 \sigma$**  significance

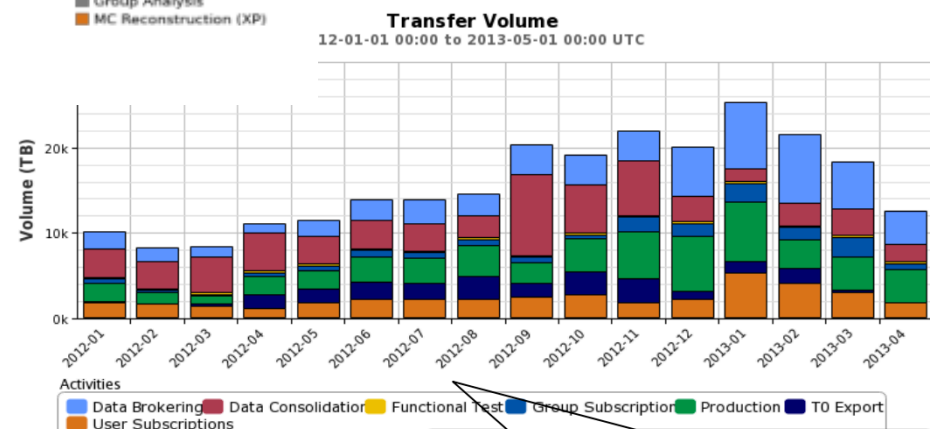


# More technically oriented ...



ATLAS: ~140k jobs running at the same time

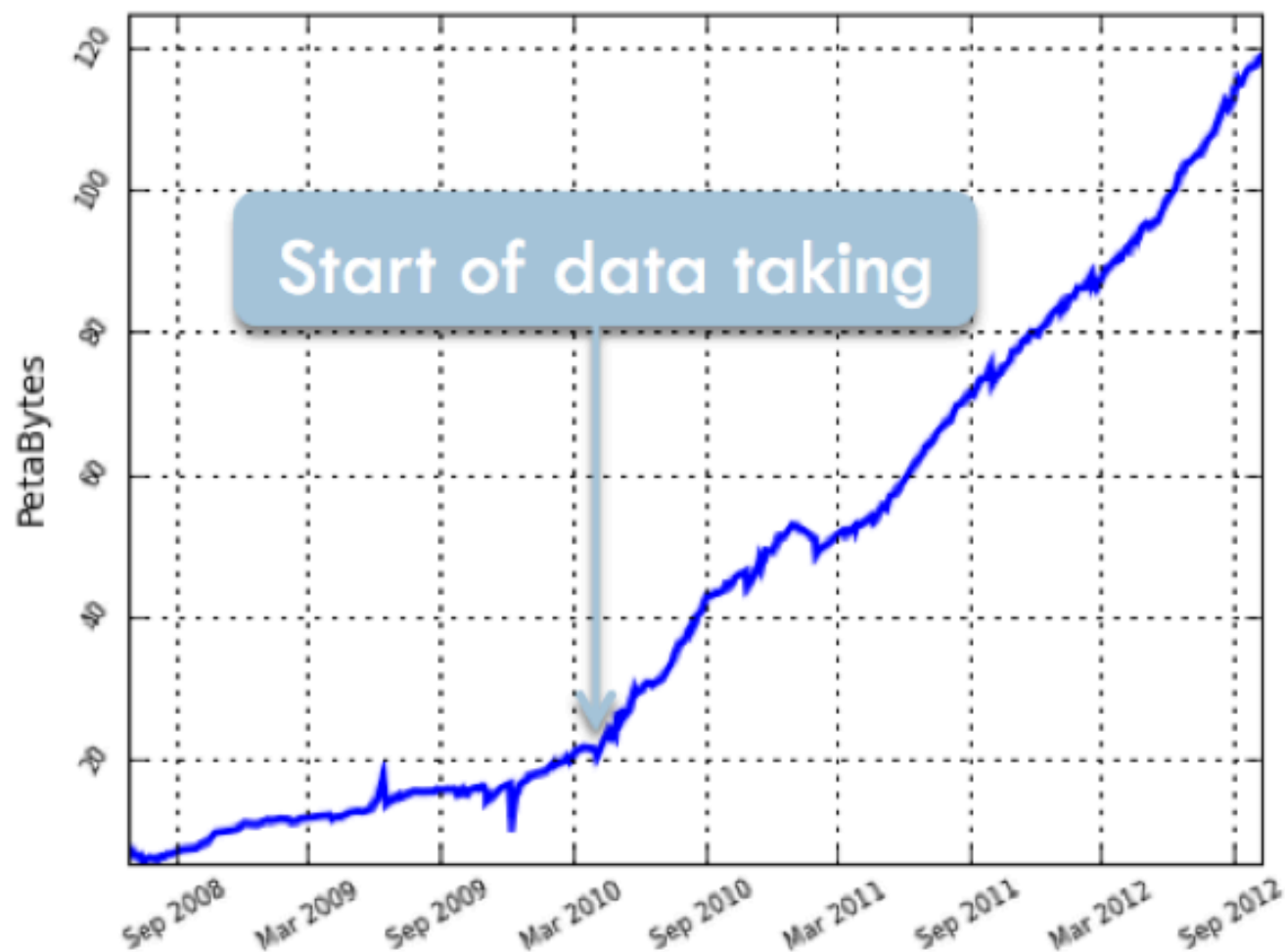
ATLAS: 30 PB moved per month



Transferred volume (monthly)



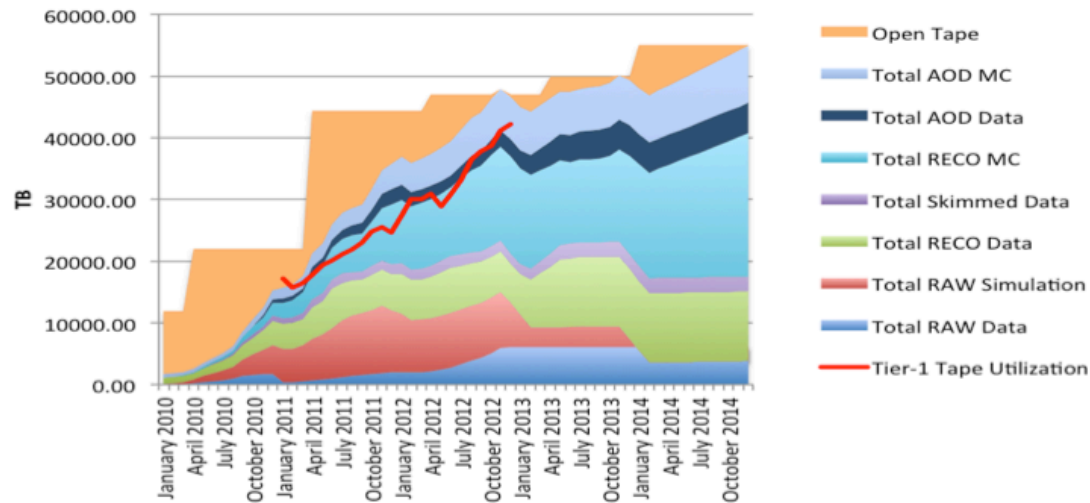
Total GRID space usage according to DQ2



**Storage as seen by  
ATLAS data  
management**

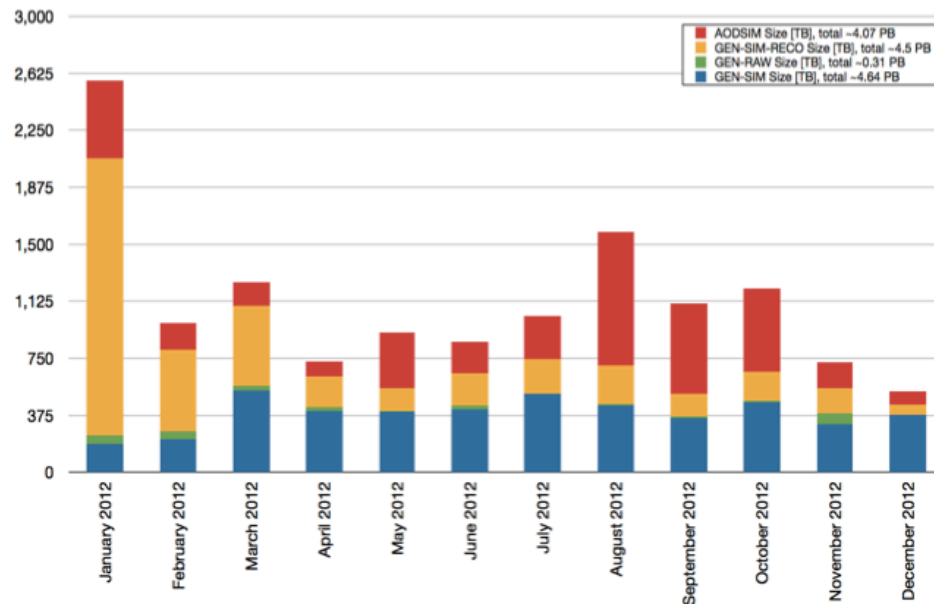
- [1] "The ATLAS Distributed Data Management project", V.Garonne et al, CHEP'12
- [2] R.Vigne, M.Lassnig at the pre-GDB on storage interfaces and access, Oct 2012
- [3] "Rucio", V.Garonne, this week's ATLAS Jamboree

## Tier-1 Tape



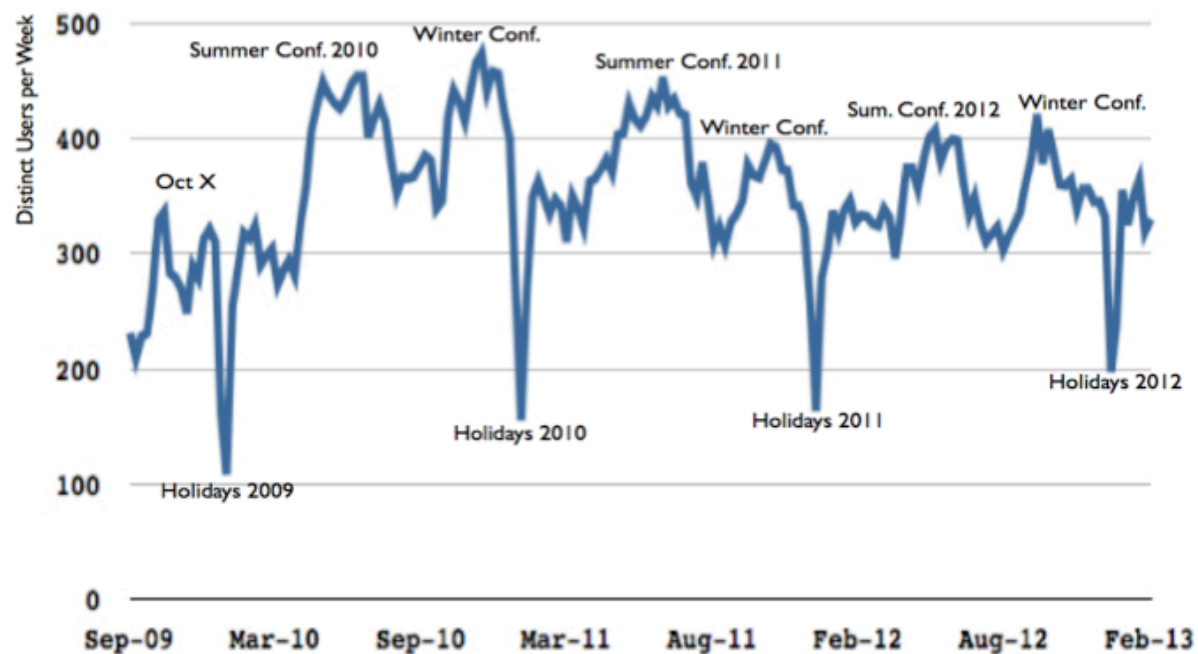
Tape usage by CMS

## MC in 2012: Size in TB per Month



MC produces by CMS:  
~15 PB/year

Figure 9: Size of the simulated events produced per month in 2012.

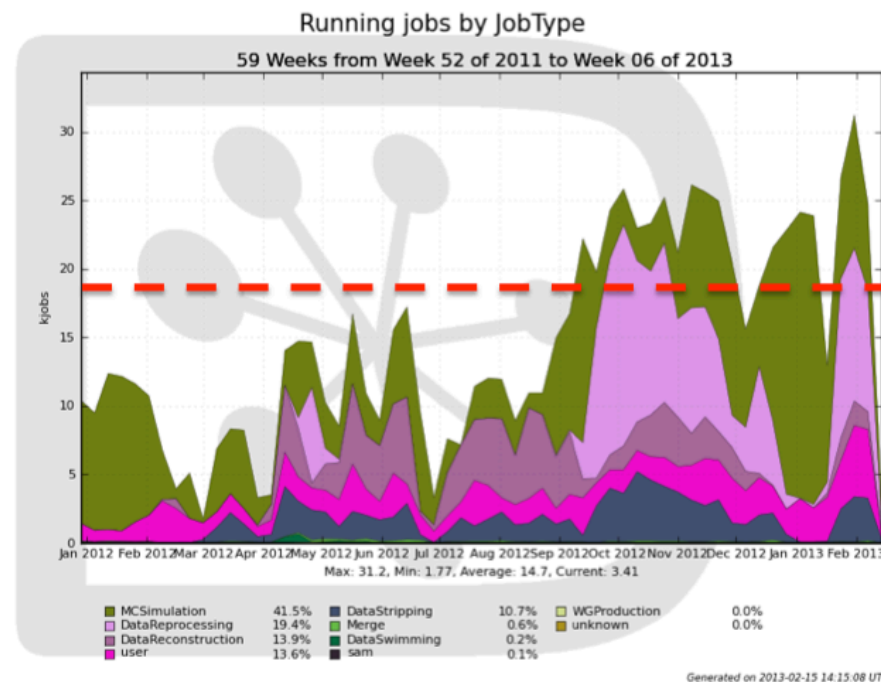


Distinct weekly CMS users in analysis

Figure 12: Individual analysis submitters per week to the grid from Sep. 2009 to Feb. 2013.

# LHCb ...

## Jobs on the GRID...



## MC production jobs

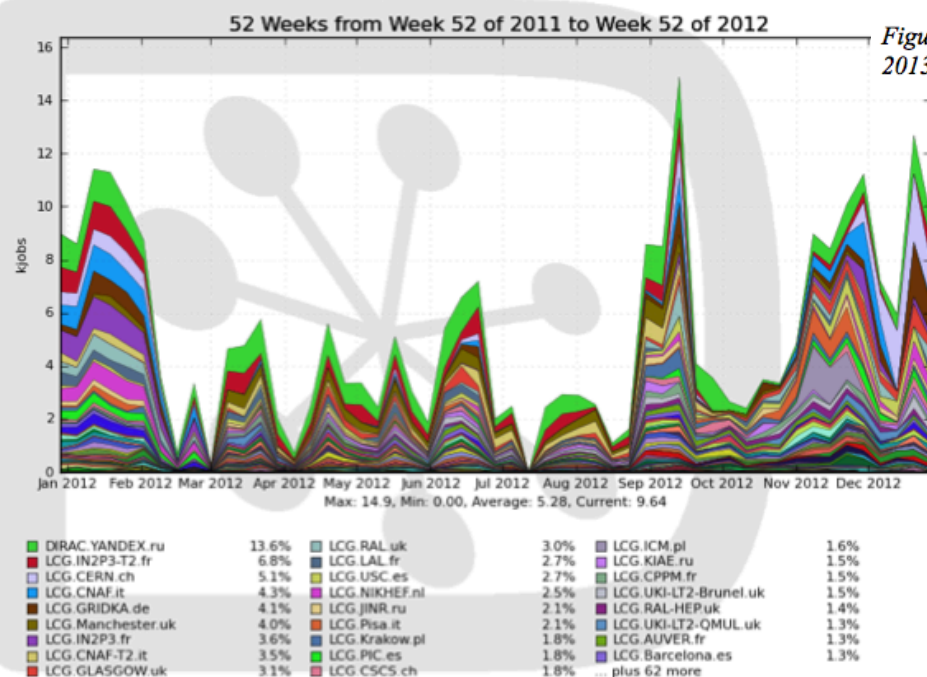


Figure 3-1: Summary of LHCb computing activities during the period Jan 2012-February 2013.

By type and by site

# The software (a small parenthesis)

- For the moment we focused on **HOW** to handle LHC computing at large scale
- We did not really clarify **WHAT** needs to be executed!
- Small outline
  - Basic software workflows
  - Overall organization
  - performance is money! The eternal fight for performance

# Basic SW workflows

- By workflow:
  - If you take today's share of Computing resources, you roughly get
    1. ~40% spent on Monte Carlo simulation
    2. ~20% reconstruction time (including Data and MC, and including the several reconstruction passes)
    3. ~40% analysis activities
- While the first bullet is mostly Geant4 processing time, on which we have not too many handles, the rest is software directly written by the Experiment
- How big/complex is it?

# A case study: CMSSW (CMS Offline SW)

- **Started development** = early 2005 (superseding an older sw)
- **Full C++** (some Fortran in externally provided routines, now gone for good)
- **A single solution** for all the use cases
  - Trigger (!)
  - Reconstruction
  - Simulation
  - Analysis
- Current size is 1120 packages, divided into 120 Subsystems

# Index of /CMSSW

Files shown: 1

Sticky Tag:  Set

## File

- [Parent Directory](#)
- [.admin/](#)
- [Alignment/](#)
- [AnalysisAlgos/](#)
- [AnalysisDataFormats/](#)
- [AnalysisExamples/](#)
- [BuildProducts/](#)
- [CalibCalorimetry/](#)
- [CalibFormats/](#)
- [CalibMuon/](#)
- [CalibTracker/](#)
- [Calibration/](#)
- [CaloOnlineTools/](#)
- [CommonTools/](#)
- [CondCore/](#)
- [CondFormats/](#)
- [CondTools/](#)
- [Configuration/](#)

## Index of /CMSSW/CalibTracker

Files shown: 0

Sticky Tag:  Set

## File

- [Parent Directory](#)
- [.admin/](#)
- [Configuration/](#)
- [Records/](#)
- [SiPixelConnectivity/](#)
- [SiPixelESProducers/](#)
- [SiPixelErrorEstimation/](#)
- [SiPixelGainCalibration/](#)
- [SiPixelIsAliveCalibration/](#)
- [SiPixelLorentzAngle/](#)
- [SiPixelPedestals/](#)
- [SiPixelSCurveCalibration/](#)
- [SiPixelTools/](#)
- [SiStripAPVAnalysis/](#)
- [SiStripChannelGain/](#)
- [SiStripCommon/](#)
- [SiStripConnectivity/](#)

## Index of /CMSSW/CalibTracker/SiPixelLorentzAngle/src

Files shown: 3 ([Show 1 dead files](#))

Sticky Tag:  Set

File	Rev.	Age	Author	Last log entry
<a href="#">Parent Directory</a>				
<a href="#">SealModules.cc</a>	1.7	20 months	mirena	Change class name to PixelLorentzAngle in order to avoid multiple declaration in...
<a href="#">SiPixelLorentzAngle.cc</a>	1.26	4 months	wmtan	Restore code backed out for previous tag
<a href="#">SiPixelLorentzAngleDB.cc</a>	1.17	5 weeks	hidaspal	trivial HVgroup, kept for use in the future

## Index of /CMSSW/CalibTracker/SiPixelLorentzAngle

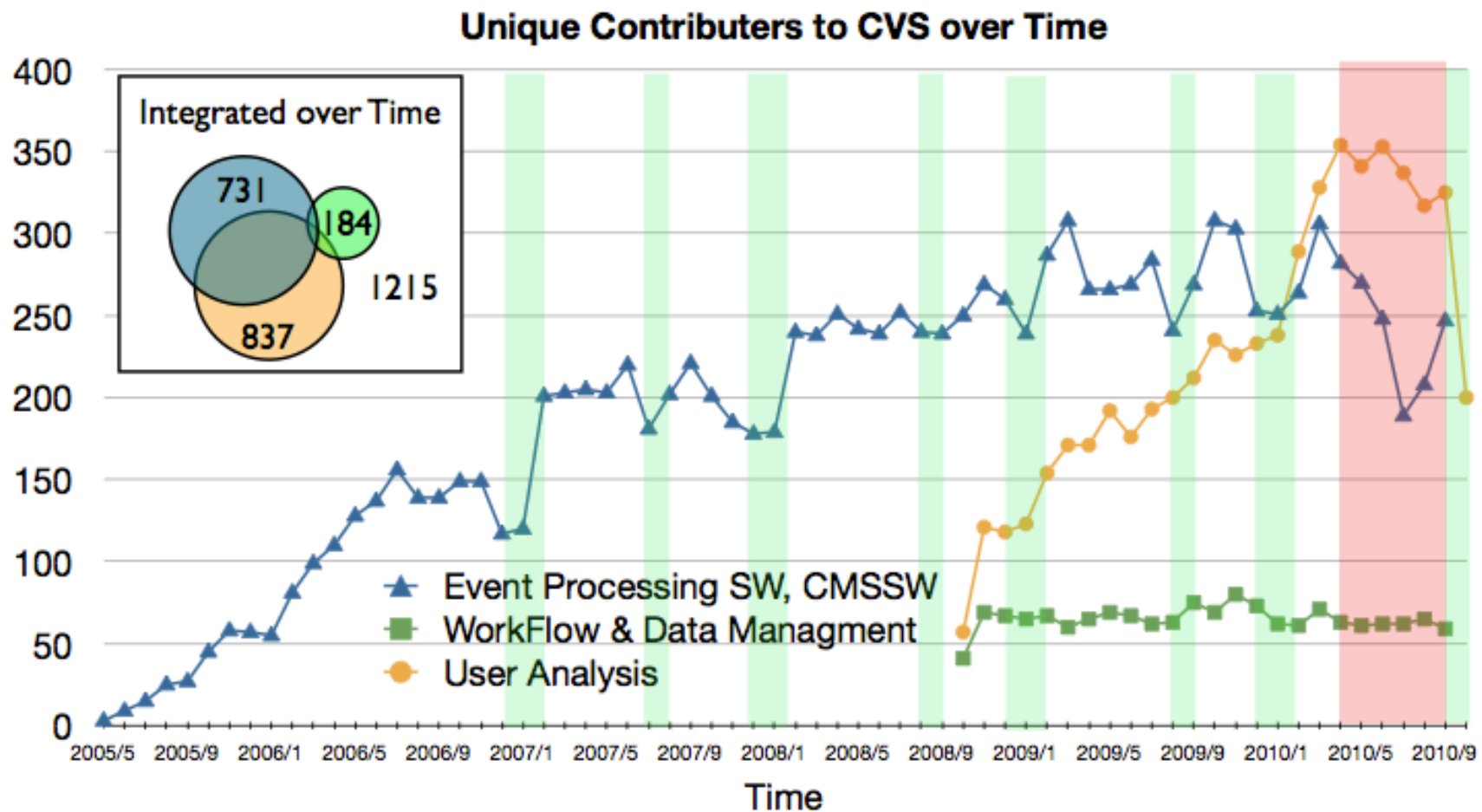
Files shown: 1 ([Show 1 dead files](#))

Sticky Tag:  Set

File	Rev.	Age
<a href="#">Parent Directory</a>		
<a href="#">.admin/</a>		
<a href="#">data/</a>		
<a href="#">doc/</a>		
<a href="#">interface/</a>		
<a href="#">python/</a>		
<a href="#">src/</a>		
<a href="#">test/</a>		
<a href="#">BuildFile.xml</a>	1.2	16 months

[Download GNU tarball](#)

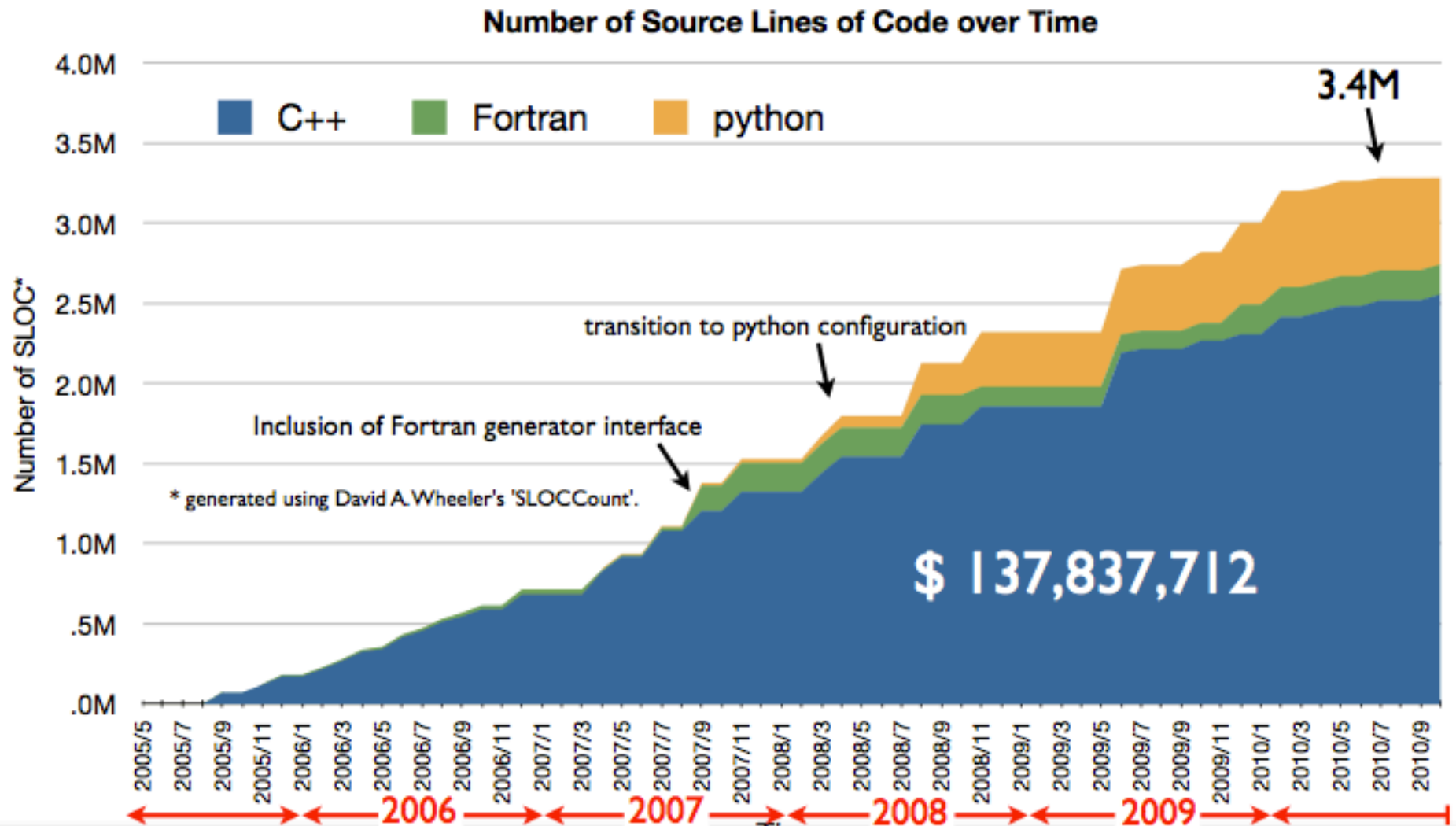




Note the seasonal variation of manpower, **vacation**, **ICHEP**

Integral is 1215 physicist contributing to the software

# Lines of code .. and \$\$!



# How difficult is to handle such a big software




















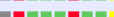
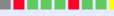













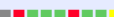
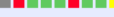



















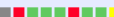
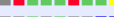








































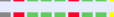
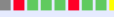

- **With ~1000 concurrent developers**
- **With > 4 M Lines of code**
  
- **You need tools ...**

# Tools for a wide scale project

- A versioning system
  - CVS, SVN, GIT
- Automatic testing facilities
  - Nightly builds, automatic tests with smart result evaluation
- Multiple releases, all traceable
  - A software installation management system

# Integration builds

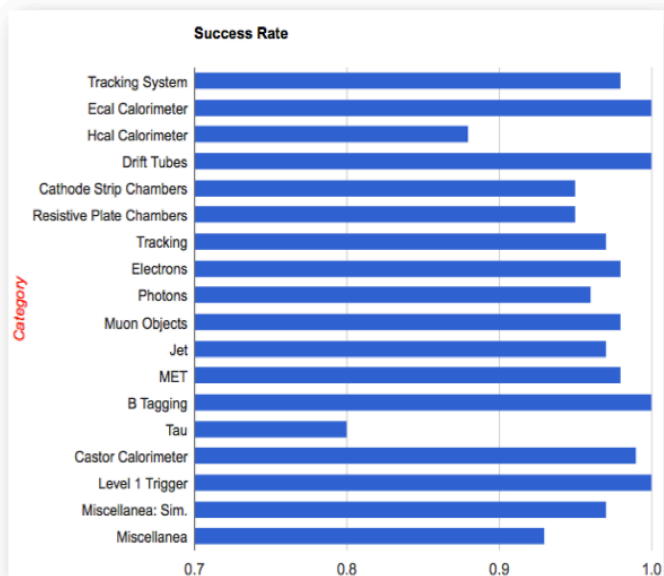
Release cycle 6.2 -- [back to top of page](#)

day	IB	platforms	builds	RelVals	OtherTests	Q/A page
mon	CMSSW_6_2_X_2013-06-03-0200	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	   <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	 <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>
sun	CMSSW_6_2_X_2013-06-02-1400	slc5_amd64_gcc472 slc6_amd64_gcc472 osx108_amd64_gcc472	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	   <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	 <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>
sun	CMSSW_6_2_X_2013-06-02-0200	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	   <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	 <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>
sat	CMSSW_6_2_X_2013-06-01-1400	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx108_amd64_gcc472	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	   <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	 <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>
sat	CMSSW_6_2_X_2013-06-01-0200	slc5_amd64_gcc472 slc6_amd64_gcc472 slc6_amd64_gcc480 osx107_amd64_gcc472 osx108_amd64_gcc472	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	   <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>    <a href="#">summary</a> <a href="#">details</a>	 <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>  <a href="#">summary</a> <a href="#">details</a>	 <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>  <a href="#">Q/A info</a> <a href="#">Strip Charts</a>

# Automatic regression system

## Reconstruction, Simulation, Calibration Constants and Trigger

CMSSW pre-releases are provided once a week in order to consolidate the state of the code, test interdependencies among software components while releases are cut approximately once per month to close a development cycle. For every (pre)release, a complete set of release validation datasets of Monte Carlo simulated and re-processed real collisions events is provided.



Summary Table

Summary	ReValH130GGluonfusion	ReValHiggs200ChargedTau	ReValUpstMM	ReValL1L1_sfts	ReValInfrBias	ReValPhotonJets_Pt_10	ReValQCD_FlatPt_15_3000	ReValQCD_Pt_3000_3500	ReValQCD_Pt_80_120	ReValQCDH135ZT_Tauola	ReValSingleElectronPt10
Summary											
AlCaReco											
Btag											
CSC											

Experts of all CMS sub-detectors, physics objects and analyses scrutinize these datasets against the ones of the preceding release to obtain an incremental validation

of CMSSW. The main tool used to perform these regressions is RelMon. For each (pre)release regression is centrally provided, with more than one million histograms compared. With this strategy, anomalies can be immediately pin-pointed.



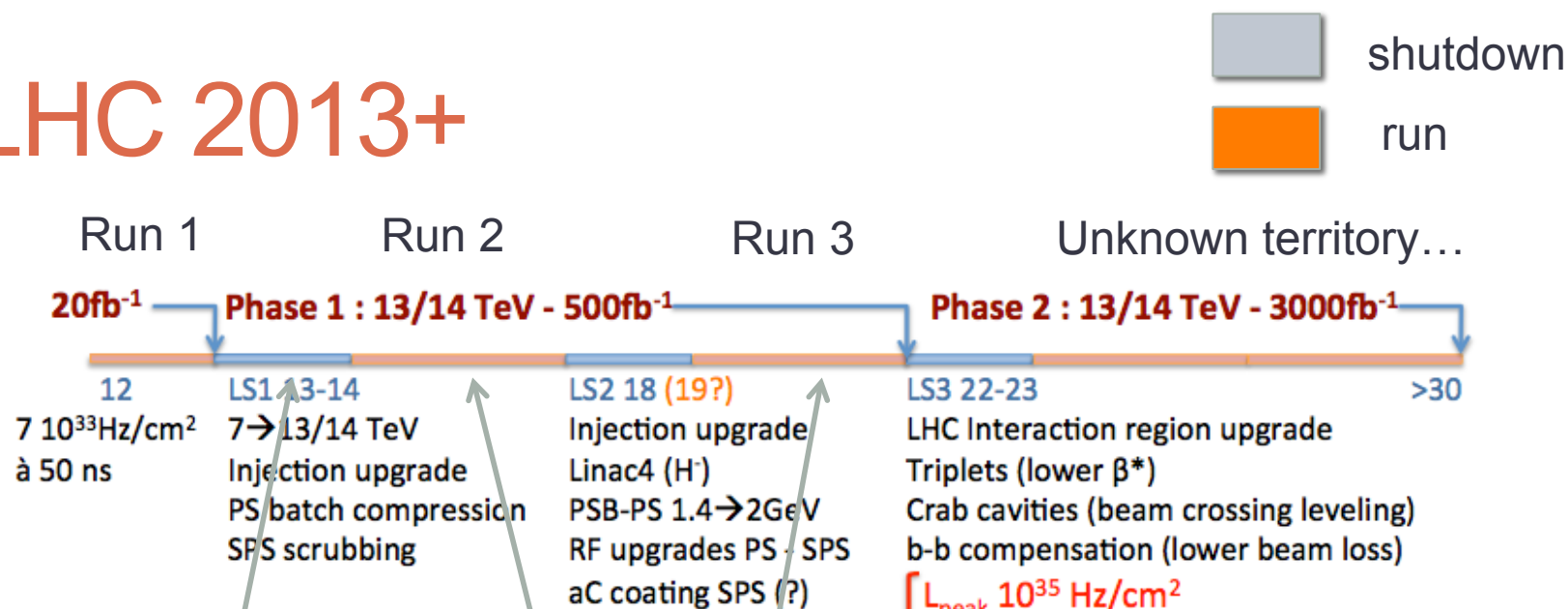
# Evolution of LHC Computing Models

## If they work (as we claim), why change them at all?

- LHC conditions are changing ... faster than technology can absorb
- We have updated priorities now (we found the Higgs!)
- Run1 Experiments had limits (due to technology being not mature)
  - We can change it now!
- BUT not to be forgotten: economical situation is Much Different now wrt early 2000x



# LHC 2013+



We are here

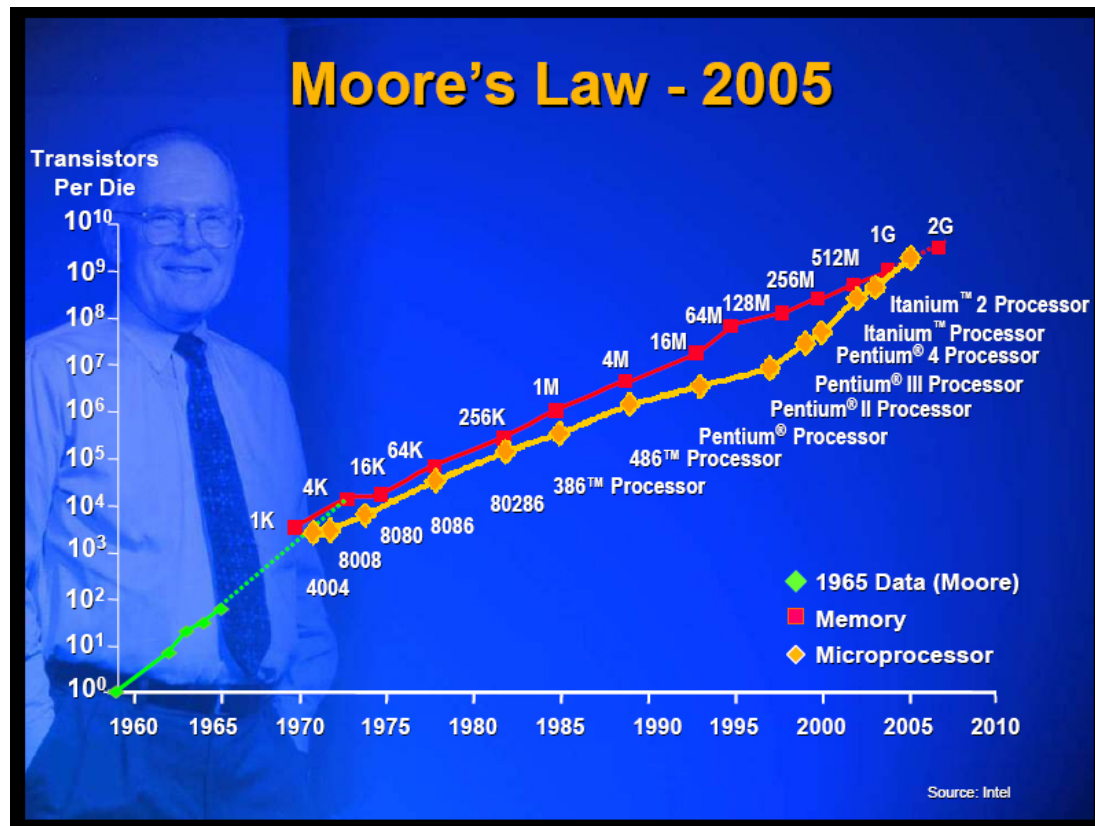
- 2015-2018: 13 TeV, ~2.5x in luminosity, up to 3x in hadronic events per collision
- 2020-2022: 13 TeV, again 2.5x in luminosity

## 2015 – how is Computing complexity going to change?

- **Greater Energy:** ~20% more particles; ~30% more cross section: **1.5x**
- **Higher luminosity:** more hadronic events per collision. LHC experiment reconstruction time is more than linear with this, so at least a factor **2x**
- **Focus on Higgs Physics:** maintain the Trigger for the Higgs channel at least at the same level as in Run 1. This requires an increase in Trigger rate from 100-330 Hz to 1 kHz – **factor ~3-5x**
- **Hence, all the rest remaining equal, we need at least a factor 10 in MORE resources (2015 vs 2013)**

# Technology does not help (enough)

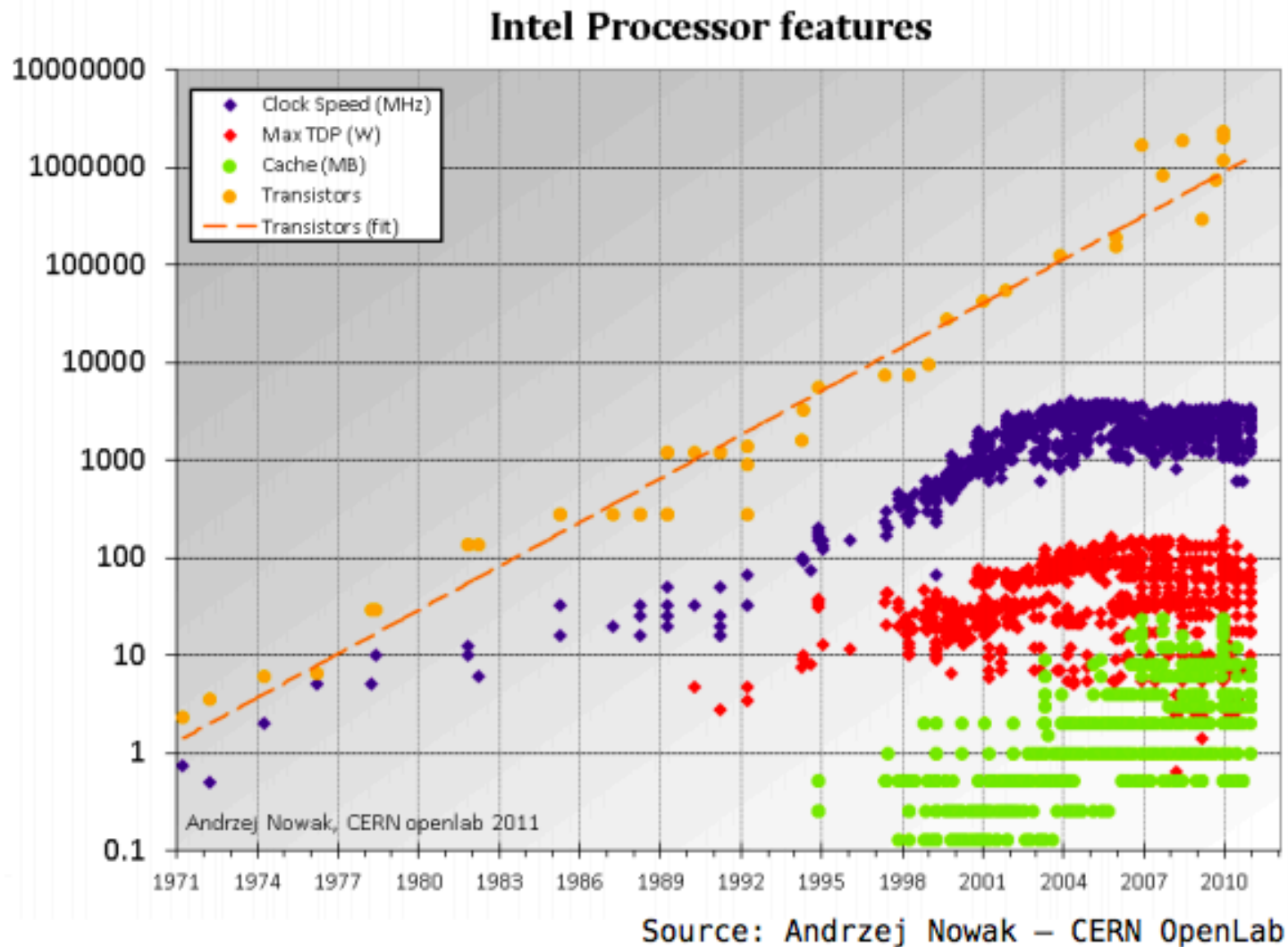
- You probably heard of the Moore's Law
- **“CPU performance / Storage / Memory for the same price doubles every (18-24) months”**



Well, but

- Apart that it is hardly true anymore (since at least 2010)
- 2015-2013 are <3 years, and it cannot account for a factor 10
- We need either
  1. More money
  2. New solutions
- (good luck with solution #1)

# Moore's law today ...



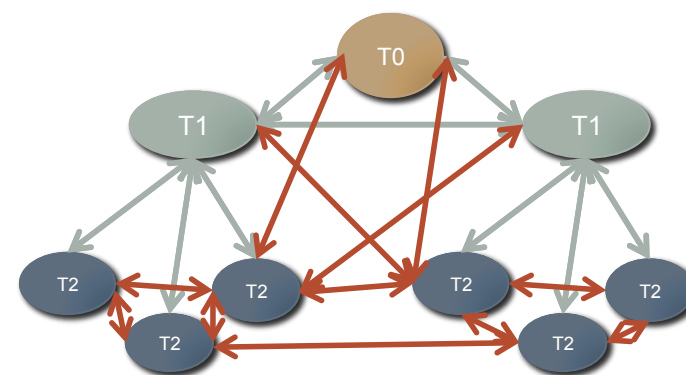
## How to gain a factor 10 (or at least a factor)

- We already said that in the GRID/MONARC approach there are some deficiencies (== lack of efficiency)
  - It is difficult (time, manpower) to move data between sites
  - Jobs go where data is means data must be preplaced efficiently
  - We have indeed a lot of difficulties in deleting data (“human viscosity”)

Solving only these could already give a factor  $\sim 2$  (for example, use the Tier1s for analysis when not busy for reprocessing activities ...)

## Already now ...

- We are partly trying to overcome MONARC
- The main problem is “solved”: networks are now much faster than the expectations
  - LHCONE (see next slides)
  - **No real need to limit (too much) traffic outside of reserved lines**
    - We already have some analysis at T1s
    - We already have data direct data moving (the so called Full Mesh)

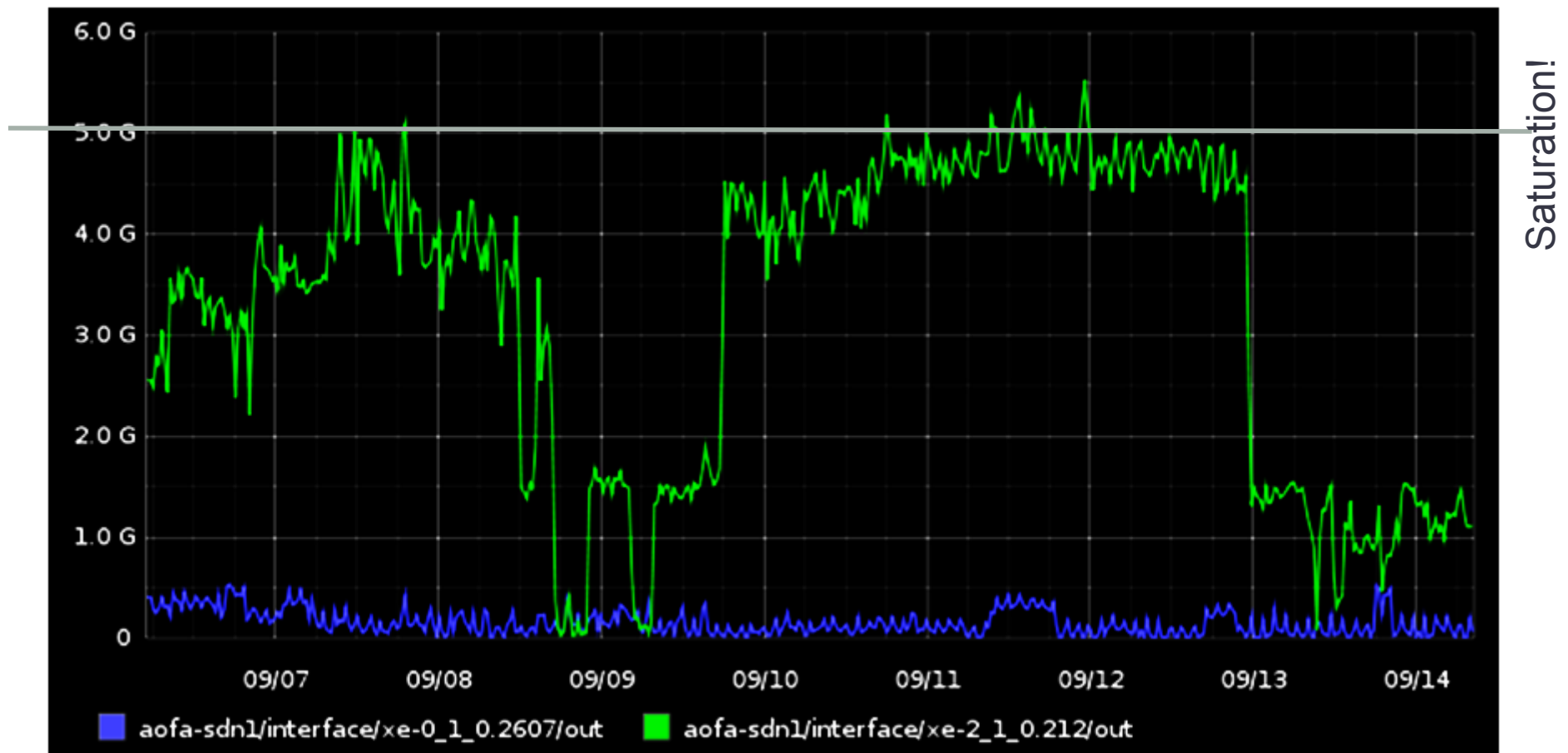


# LHCONE !

- Reminder: LHCOPN is the network between T0 and T1s
- T2s are served via standard NREN links
- But these links are now much better than expected
  - A US T2 is already linked at 40 Gbit/s, soon to become 100 Gbit/s
  - An Italian T2 is now at 10 Gbit/s, soon to become 40 Gbit/s
- Such a capacity is USABLE by the experiments, which have started to trust in a full mesh of  $50^2$  sites
- **And then what is LHCONE? Simply a “protection” for the rest of the (research) world!**

# The Need for Traffic Engineering – Example

- GÉANT observed a big spike on their transatlantic peering connection with ESnet (9/2010) coming from Fermilab – the U.S. CMS Tier 1 data center



- This caused considerable concern because at the time this was the only link available for general R&E



## LHCONE (2)

- It is a virtual network linking all the LHC T2s.
- Virtual = it does not have private fibers like LHCOPN, just uses NRENs
- Can be forced to stay within its limits, without cannibalizing the rest of the research traffic
- **Within this limits, even if not “reserved”, you can basically count on it not too differently than LHCOPN**



# How to use the new network facility?

- **Direct Remote data access (a.k.a Streaming!)**
- You remember the problem with DataDriven: 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!
  - You do not download the file, you see it over the network

# Storage Federations

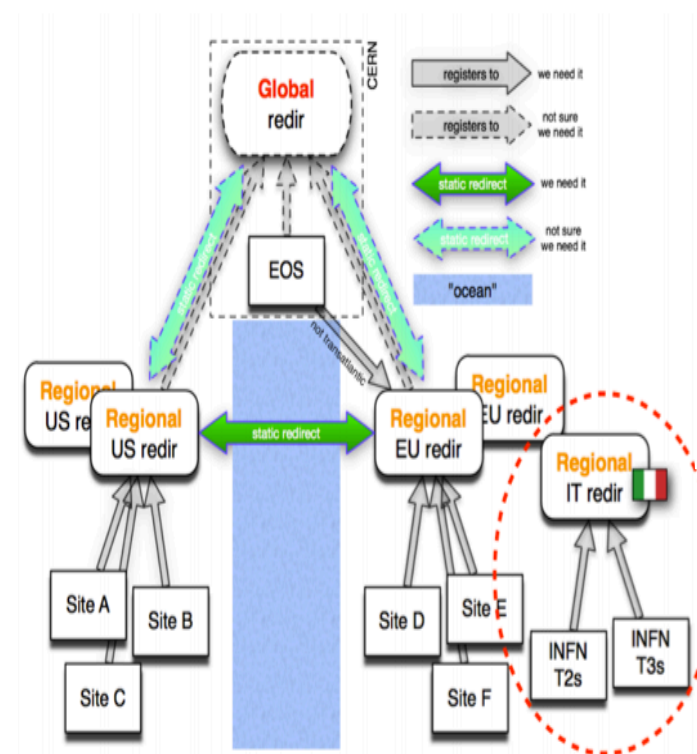
- Imagine the scenario:
  - You put data anywhere (on any of the Sites serving the experiment)
  - Jobs go anywhere CPU is available
  - Jobs have to access data:
    - How? Via a remote access protocol
    - Where from? It would be better from a close place
- Storage Federations are a way to fake the existence of a single storage system, and to implement priorities

# Storage federations

- Recipe:
  - Distribute data on N sites
  - Choose a remote data access protocol (http, Xrootd, NFS4.1 ...)
  - Build a file catalog / a redirection service
    - The first being a DB (bleah..), the second a fall-back option
- Access data directly from the “federation” in a multi hop scheme... see next slide!

# Idea: federazione gerarchica

- **(Examples: FAX, AAA)**
- When a file is opened (POSIX fopen)
  - If the file is local (local storage), open it; otherwise
  - Ask your national redirector. If the file is found in your country, open it; otherwise
    - Ask you regional redirector. If the file is found in EU, open it; otherwise
      - Reach the top level redirector; if the file is found, open it, otherwise -> ERROR
- While all the files are accessible in this way, “cheap” transfers are tried at first



## Sounds it works, but which is the risk?

- Xrootd, Http are very efficient protocols,  $O(1000)$  such connections can saturate ANY link
- It is essential to “shape” the traffic, not allowing Xrootd to hit too heavily a Site
- It is essential to have a good data preplacing, such that most of the “fopen” are resolved locally
  - You still need some sort of data preplacing, sorry ...
- (Unless network capacity becomes infinite....)

# Financial problems ahead...

- Most of the (EU side) GRID project, including WLCG, are basically gone (end of FPVII European Program)
- At the same time, most funding agencies are having difficult times (research budget going down and/or spent on other researches)
- We are not the biggest users of Computing in research anymore! A few examples
  - [Kilometer Array telescope](#) (circa 2025: 2900 antennas, spread into multi million squared kilometers): **multi terabit** signal delivery on O(1000) km, final data for analysis **exceeding 100 Gb/s**
  - [Human brain project](#) (>2020):  **$10^{18}$  flops**, ~10M today's cores (or 1M GPUs)



# External resources

- We cannot continue assuming we will be given the resource we ask for.
- We should be able to use external resources (computing facilities for other sciences, supercomputing centres, free Amazon time) as soon as they become available
- With MONARC it is not easy at all: **a generic “computing centre” is much different from a Site**
  - No GRID MW
  - No experiment support
- How to do it??

# The Cloud!

- A general answer to the previous question is the **Cloud Paradigm**
- Seen as an evolution to the GRID:
  - While the GRID wanted to port applications on Distributed Systems
  - The Cloud wants to move full Infrastructures to it
- So, not just the final piece of code you want to run, but the whole computing environment becomes geographically distributed and virtualized

# Let's try and clarify the difference with an example

- Today when you submit a computational task, you assume you are landing to a Site which “knows” how to handle it
- This means there is someone there who made sure
  - The version of the operating system is fine
  - You have enough RAM
  - Your experiment software has been installed there
  - You have (some) data on the local storage
- So, if someone gives you 10000 CPUs for free in a random place, **you simply cannot use them without notice**

# How a Cloud approach changes this

- Usually Clouds use
  - A **virtualized approach** (think of VMware): you are not running on the system itself, you are running in a virtual machine
  - You expect not to send just your computational task, **but the full environment** (a virtual machine already configured for your experiment)
- +
  - You are expected to access data from the Storage Federation: **no local storage needed**
- ...

# GRID vs Cloud

You need	GRID	Cloud
A computer connected to network, with conditioning and power	<b>Local site staff</b>	<b>Local site staff</b>
An operating system “compatible” with the application	<b>Local site staff</b> , after negotiation with experimenters	Comes as a virtual image from the <b>experiment central infrastructure</b>
A local installation of the experiment software (and a local area where to store it)	<b>Local site staff</b> provides area, <b>Experiment support</b> installs software (ok, we have CVM now ...)	Downloaded on demand from the <b>experiment central infrastructure</b>
Machines for local experiment facilities (voboxes etc)	<b>Local site staff</b> provide them	Also deployable as virtual images
A local storage containing the input data	<b>Local site staff</b> needs to have bought storage for the experiment	Data can be accessed remotely, or via cloud storage (efficiency, price?)
A configuration to be executed	<b>User!</b>	<b>User!</b>

## So, Cloud help us to use the “opportunistic computing” ....

- This means using computing resources you were not expecting to use, like
  1. **Access resources not specifically built for your use (like computing resources from other sciences)**
  2. **Use resources you bought, but were not expecting to use in this way (like the trigger computers, basically idle when LHC is not running)**
  3. **Commercial resources (Amazon EC2, Google, RackSpace) you access via .. your credit card!**

# Some examples

1. In US many supercomputing center (Argonne, SanDiego, ...) are keen to offer some CPU time to LHC
  - Given their scale, “some” may mean a 2x for us!
  - But, they use
    - a specific linux distribution
    - A very strange local networking
    - They do not want to offer storage
  - You simply cannot deploy a GRID there, but a Cloud is viable!
2. A relevant fraction of the computing power LHC bought resides in the Trigger Processors. They are not used
  - When LHC is off (either for long periods, LS, or a few hours, when they refill the machine)
  - These machines do not even have a batch system, are configured to run a single task
  - Again, you simply cannot deploy a GRID there, but a Cloud yes!
3. We did not invent the Clouds, most implementations come from commercial entities
  - Can we use them?

# Just an example: what if ...

- ... we decided to use **ONLY** Amazon instead of our centres?
- Some estimates:
  1. CMS: move 1 month of Geant4 processing to Amazon with standard pricing is ~1.3 M\$
  1. ALICE: move ALL 2012 computing offline activity to Amazon

35000 concurrent jobs (~50000 at peak)	\$ 1.35 - 1.92 M/month
12 PB on disk	\$ 1.24 M/month
14 PB on tape	\$ 0.14 M/month



- **40M\$**

---

	\$2.73 - 3.30 M/month
	\$32 - 40 M/year



# It is not currently viable

- Staying with the ALICE example, the budget member states pay for that is not entirely clear, but should be at least a factor 4 less (not clear since you do not know how to account for salaries etc)
- (and ALICE is quite smaller than CMS and ATLAS)
- So at the moment the only real use case is
  - I “need” more resources now (in this way experiment X will be able to publish before experiment Y and we will win the Nobel Prize)
  - Some institution uses its “credit card” to shell out 1 M\$ for that
  - To my knowledge, did not happen (yet)

# But ...

- In recent months a commercial alternative seem to have risen: “spot market”
- **The figures in previous slides are for standard Amazon usage (I reserve CPUs and I use them).**
- But if I move to:
  - I buy 1 Million Hour of CPU in the next year, allowing Amazon to decide when to let us use them (using moments when their load is low)
  - I even allow them to shut down my jobs when something “paid better” arrives
- I can reach a **factor 10 of discount!** What I lose is basically the ability to run “now” and not “tomorrow”
  - Again, no one ever used it for the moment apart from tests. But can be economically viable!
  - It cannot be “all of our computing”, given the time restriction, but can be ok for long range Monte Carlo production

# What is the gain

- By using
  - Full mesh Computing model, where activities can be carried out ~ everywhere (not in a specific Tier only)
  - Use opportunistic computing
- We expect to lower the computing requests by a factor ~2x
- ... **A factor 5x still remains...**
- For that: software /architecture improvements!

# Our computers up to now

- We use pretty standard out of the shelf computers
- Today you can buy for ~7000 Euros
  - 64 computing cores (x86\_64)
  - 256 GB RAM
  - 8 TB disk
- On this, we use to run 64 “jobs”, independently (a farm, not a cluster), with the performance described yesterday



A “thing” like this is ~

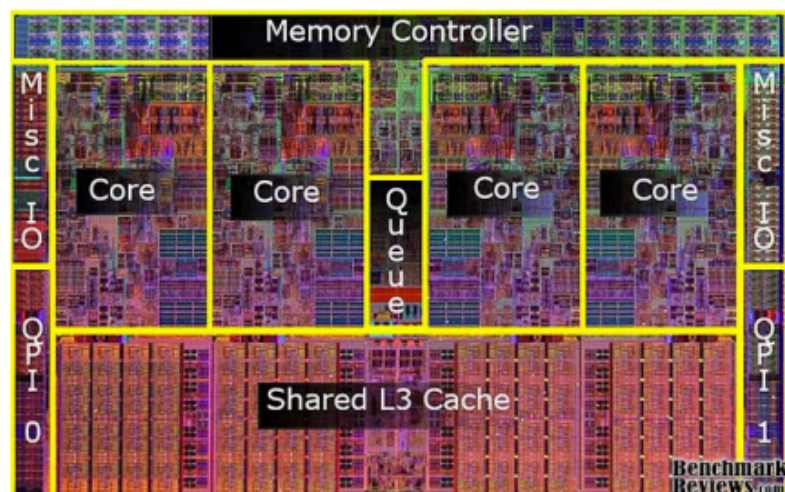
- 700 HS06
- Consumes 1 kW + 500 W for cooling
- Has a lifetime of 3 years
- It costs 4.5kEuro on power in these 3 years

# What's (sort of) new in computing?

1. Multicore processing: treat one such machine as a single job instead of 64 distinct machines
  2. High performance vector units: Xeon Phi, GPGPU
  3. Low power architectures (ARM...)
- Let's say a few words on them



# Multicore Scheduling



Since their introduction in ~2005, we have been scheduling ~one job per core on multicore CPU's.

This works and allowed throughput to continue to scale, but is not optimum for optimizing use of these processors.

The Framework and CMSSW are being prepared to exploit multiple cores from a single cmsRun. During LS1 will switch to scheduling “multicore” jobs in all grid sites.

---

In the first approximation the initial switch to multicore scheduling will **not** increase total throughput. We should be able to significantly reduce memory requirements and stabilize many scaling aspects (#jobs, #files, etc.)

---

In principle the number of running “jobs” in the system would drop from ~80k to 10-20k and will stay roughly constant going forward. (Production will run multicore, analysis will run single-core, but be scheduled within multicore job.)





# Multicore scheduling

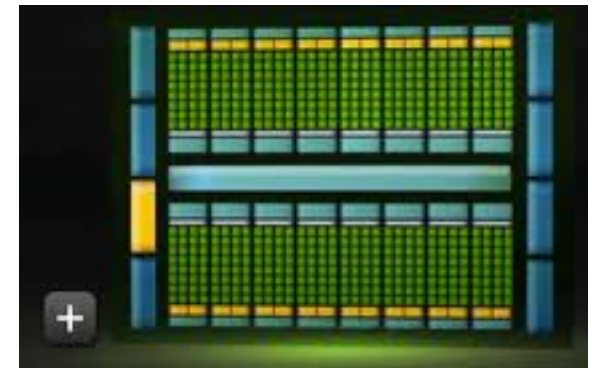


Even if the deployment of the multicore framework does not immediately increase throughput, finally pushing through the transition to multicore scheduling *does* open the door to possibilities subsequent gains:

- deployment of fine-grained parallelism within the code (perhaps with changes to data structures, etc.) will utilize the memory caches better
- the fact that we take scheduling multiple cores into our hands should allow us eventually to co-schedule CPU-intensive jobs (e.g. simulation) when the primary job(s) are not fully utilizing the CPU's
- Eventual self-tuning of the application itself? (e.g. for exploitation of Hyperthreading)
- Moving from “memory-constrained” to (slightly) “memory rich” might afford other opportunities for improvements. Reduced memory requirements facilitate also some opportunistic use.

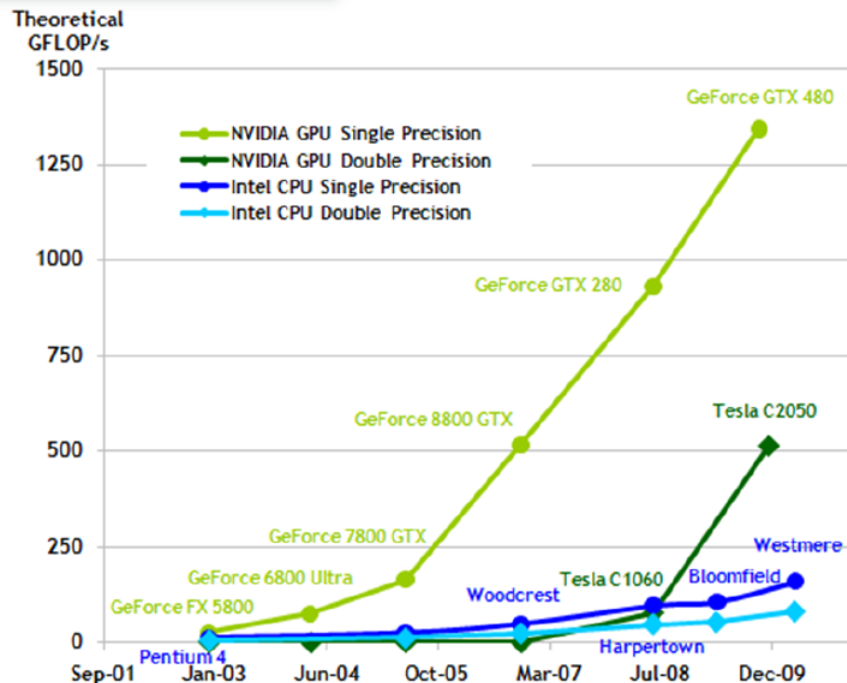
# New Architectures (1)

- Massively parallel CPUs are with us since at least 5 years
  1. General Purposes Graphical Processing Units (GPGPU)
    - Video games oriented Graphics Cards recycled as Vector machines
    - Up to 1024 cores per board
    - Vector processing = they are only able to repeat the same operation on multiple data (Single Instruction Multiple Data = SIMD)
- Very powerful, but SIMD is limited to very specific applications (matrix multiplication ... and eventually particle propagation)





# GPGPU – relative performance



A high end GPGPU

} Intel x86 cores

But beware:

- Very power hungry
- This kind of performance just for very specific use cases
- Very difficult to program



# Intel Xeon Phi



Xeon Phi 5110P  
8GB Memory,  
60 Pentium-derived  
cores @ 1.05GHz,  
512bit vector units

Intel MIC architecture, currently packaged as a HPC-oriented coprocessor (card sits in a PCIe slot of host and runs an independent instance of linux).

Several possible usage modalities: “batch” use with application running directly on the coprocessor, “offload” use with specific calculations running on the coprocessor. Effectively a heterogenous mix of host and coprocessor.

These are becoming available this spring in various sites for tests.

## Xeon Phi (2)

- The main advantage is that it is still x86, so no new programming technique needed
- But
  - Also very power hungry
  - Its performance in real life have yet to be proven

## ARM (3)

- A low power architecture (so attacks per price problem from another side)
- Still much less performing than x86\_64 (at least a factor 4 less)
- But per Watt, a factor 4 better!

x86	x86	ARM	Type	Cores	Power	Events/ min/core	Events/ min/Watt
			Exynos4412 Prime @ 1.704GHz	4	4W?	1.14	1.14
			Xeon L5520 @ 2.27GHz	2x4	120W?	3.50	0.23
			Xeon E5-2630L @ 2.0GHz	2x6	190W?	3.33	0.21

CMS test (ARM vs x86) with simulation (geant4)

- Events/core/min still worse
- But Events/min/Watt largely better
- Would allow construction of much cheaper computing centres
  - Much less in \$\$ per power bill
  - Much less cooling infrastructure

# Nice and small machines

## Odroid U2



### A Samsung Galaxy S3 without screen ...

- Exynos 4412 4-core @ 1.7 GHz (ARMv7)
- 2 GB RAM
- GPU Mali 400 4-core
- eMMC memory (64 GB)
- 5W idle, < 7W when under load
- Below 100 \$

## Boston Viridis



A cluster in a single box ..

### In 2U:

- 48 SoC ARMv7(1.4 GHz) 4-core, each with 4 GB RAM
- 8x10Gbit/s internal networking; 24x DISK slots
- Under 300W under load
- ~ 10kEuro (?)

# All in all ...

- We expect to be able to squeeze another factor x2 by using
  - Multicore chips in the correct way
  - New architectures when possible (GPGPU for tracking, ARM to lower power bill)
- We were 10x off, we would gain 2x from Cloud/Operations, 2x from architectures
- We are still a factor 2 off, which we hope can be partially solved by Moore's law
  - No magic solution here for the moment

## And after (> 2020) ?

- No clear picture available at the moment (we are still trying to digest 2015-2020)
- **Some nice ideas:**
  - **Have all the trigger running via standard PCs** (no dedicated electronics) – so feed 40 MHz directly to linux farms
  - **Do not save anymore RAW data, but directly analysis data** (would allow a 20x more rate to disk)
    - **But you need a really strong faith on online calibrations!**
  - **Use only remote access** (no preplacing!) – you really need to live in a “infinite network” situation
  - **Use Clouds and not GRIDs for everything**, offload a great part of the computing to commercial entities
  - **Use GPGPU/Phi for Trigger, ARM for anything else**
    - **Ease the construction of “light” computing centers, with no/less infrastructure**

# Conclusions

- Computing models for HEP have allowed current experiments to
  - Take data
  - Process them
  - Publish
- So it is hard not to consider them a success story
- To date, the biggest (public) effort in Computing
- Hard times ahead, we cannot relax: next generation of experiments/upgrades are already putting the models to their limits
  - But he do have ideas!



# Some PR ...

- Among the activities you can research on in HEP today, this is for sure
  - The one which opens more doors in industry later
  - The one in which you can have more fun (if you are the nerd type of person)



**WE WANT YOU!**