Software and Computing Challenges

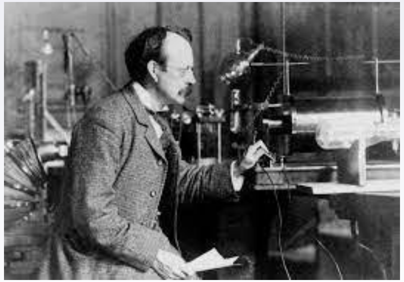
Tommaso Boccali – INFN Pisa

Outline

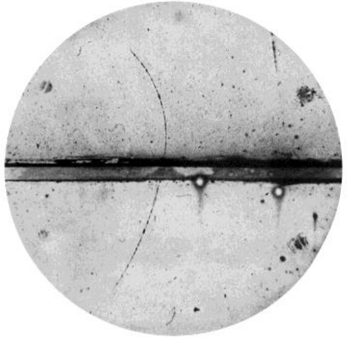
- Why this talk?
  - Why are computing and software important / an issue for High Energy Physics?
- The current picture – how it is working
- Expected evolution of needs in the next decade(s)
  - Is that a problem? Can we cope?
- The current (most notable) trends in Computing for HEP

Why is computing a relevant aspect in HEP

- In High Energy Physics, the era of low hanging fruit is long gone
  - In the first 30 years of 20th century, a tabletop experiment and maybe a photocamera was enough for groundbreaking studies
- Now we are in the regime where in order to be relevant one needs to look into **high energy** events and/or **rare** processes and/or **very precise** measurements
- In all these 3 cases, the need for a lot of computing is



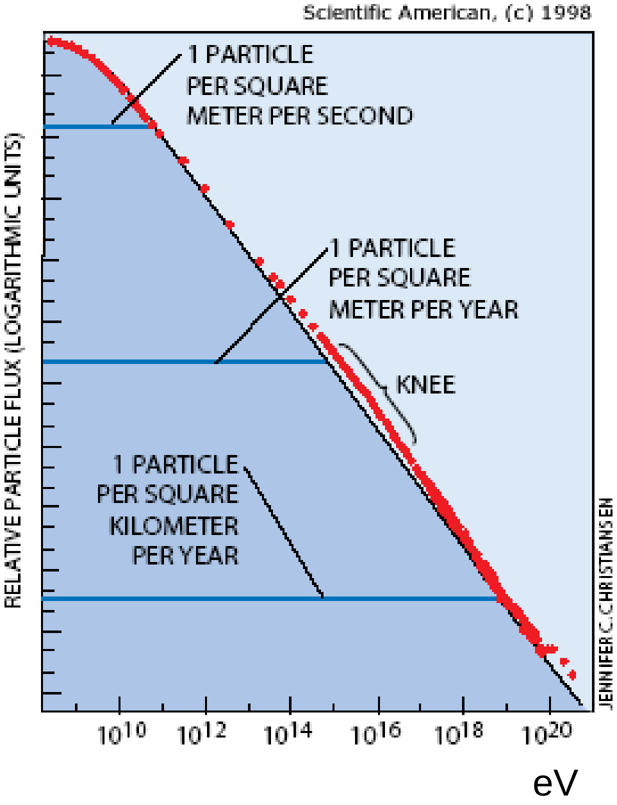
Thomson, e$^-$, 1897



Anderson, e$^+$, 1932

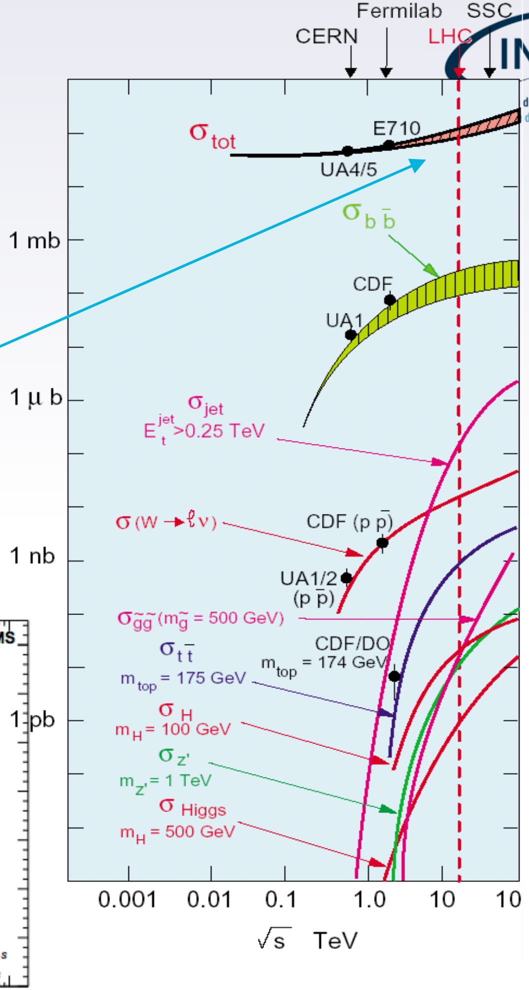
How to?

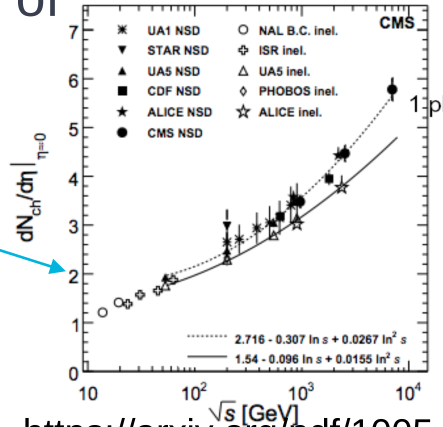
- High Energy: Look up in the sky!
  - Astroparticle Physics, the universe produces for you cosmic rays (measured up) to some 10^{21} eV (10^{9} TeV)
  - But they are rare!
- Rare: Produce (a lot of) high energy events using colliders
  - Current best is “only” at 13 TeV (c.m.), but we can produce billions per second



... but why?

- Most of the reasoning involves the relation between the cross section of a given process and the number of events generated
  - **$N = \sigma \times L_{int}$**
- More (c.m.) energy in the collision of beams: the total cross section increases + the complexity of the collision results increases
- **You get more events, and more crowded**

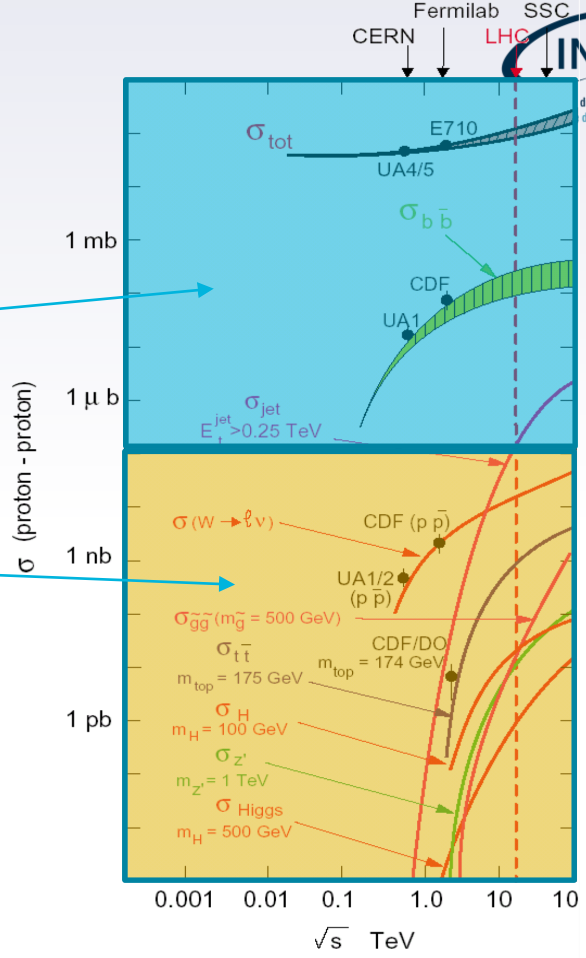




https://arxiv.org/pdf/1005.3299.pdf

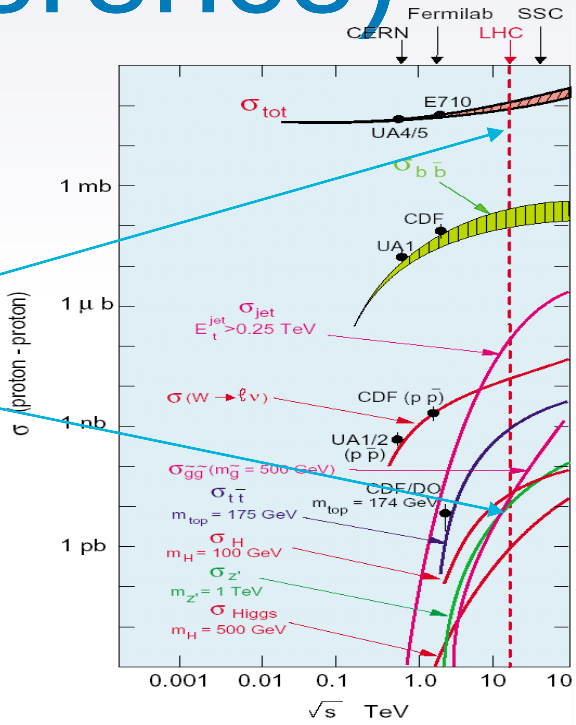
… but why (2)?

- This part of the cross section plot is “mostly understood and not interesting” (we do not expect to extract easily new knowledge from it)
- 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, you need to produce (a lot of) the former**



Some numbers (CMS and ATLAS used as a reference)

- ATLAS and CMS: general purpose, but certainly designed with the Higgs discovery (or non-discovery) in mind
- So you want to study a process **here** but to do so you cannot avoid to generate (a billion times more) **uninteresting processes**
- But how many “trials” you need?
  - assume you know the Higgs prod cross section is somewhere **1-100** pb, and the total cross section ~ 100 mb



Total number of “trials” needed

- If your goal is to have 10.000.000 produced Higgs in 5 years (per experiment):
- **L_{int} = 100 fb^{-1} (1e7/(100000 fb))** and then, scaling to the instantaneous lumi (assuming an efficiency factor ~5 for shutdown periods, vacations, repairs, etc), when you remember that 1 b = 10^{-24} cm^2 → L_{int} = 100 fb^{-1}=10^{41} cm^{-2}

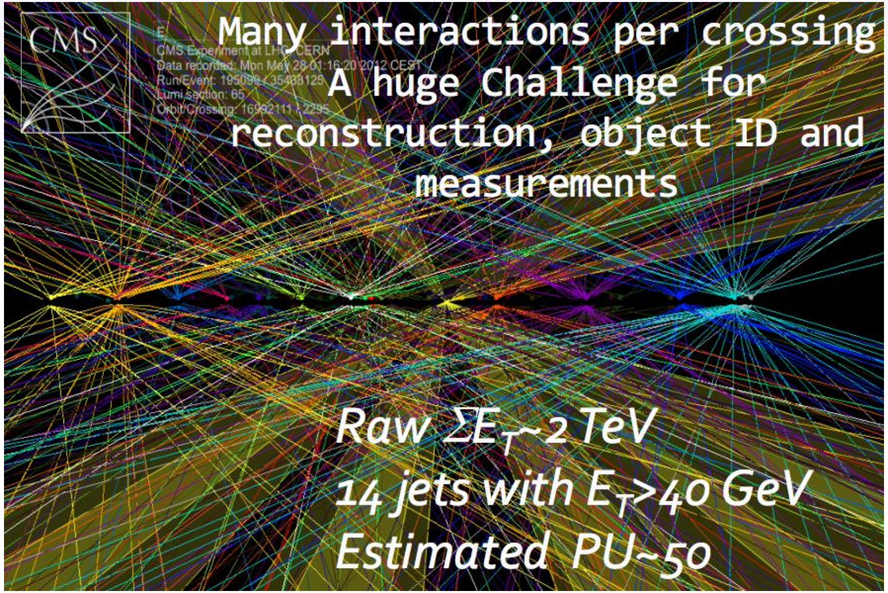
  The LHC!

  - **L_{INST} = $5_{(ineff)}$ * 10^{42} cm^{-2} / (5 y *3*10^7s/y) = O(10^{34}) cm^{-2} s^{-1}**

- **.. But at the same time, 100 fb^{-1} will generate some 10^{16} «uninsteresting» collisions**

… well but … I can select only the interesting ones!

- Not an easy task, they do not always look so different
- On top of this, the 25 ns bunched structure of LHC (linked to the capability of beam injection, ond to the capability of our detectors to discriminate events only if they are “distant in time”) superimposes events (~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!)
  - An online selection is not trivial; in order to have decent efficiency on the “interesting events” you cannot be too picky
  - For some areas of physics (the B sector, for example), even the interesting events are _a lot


CMS
CMS Experiment at LHC, CERN
Data recorded: Mon May 28 01:16:20 2012 CEST
Run/Event: 195099 / 35438125
Lumi section: 65
Orbit/Crossing: 16992111 / 2295

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

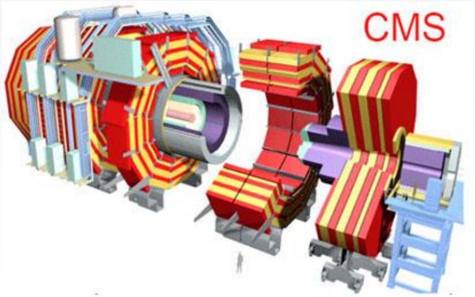
Raw ΣE_T~2 TeV

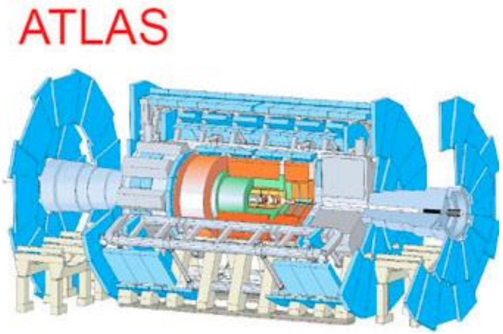
14 jets with E_T>40 GeV

Estimated PU~50

Let’s do a back of the envelope estimate of the storage needs

- We can use a simplified “IT” model for “a detector”
  - It “takes a picture” of a collision event every 25 ns (@ 40 MHz)
  - It has ~ O(100) Million acquisition channels (10x for the detectors to come)
  - Assuming 1 channel = 1 byte, the virgin data rate would be →
- **40e6 ev/s * 100e6 byte/ev = 4 PB/s**
- A“storage problem” is automatic given the needs for looking into rare events with an high





The storage

- 4 PB/s in 5 years would be 120 ZB (ZettaBytes! ; 1 ZB = 1 Million PB = 10^{21} bytes) →
- Of coarse 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
- I am not detailing them here, but some of them:
  - **Easy ones**: Zero suppression: do not save the reading of channels which are not “significant” (lossy compression): 100 MB/ev → 1 MB/ev
  - **Complex ones**: try and interpret the events as they flow, and select “enough of the interesting ones” → **the trigger** (not covered here, sorry)!

Storage (and CPU) drive the trigger rate

- In an ideal world, all the 40 MHz 25 ns snapshots (events!) would be saved and analyzed
- In practice, a much lower rate can be 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 the end, it is a tension between what you can afford and what you would like to collect; LHC history (CMS-ATLAS) is
  - Run-1 (2010-2012) : 100-500 Hz (out of the 40 MHz)
  - **Run-2 (2015-2018) : ~ 1 kHz**
  - Run-3 (2022-2024) : 1-2 kHz
  - Run-4 (2027+, see later) : > 5 kHz

“what is the limiting factor @ a HEP experiment?”

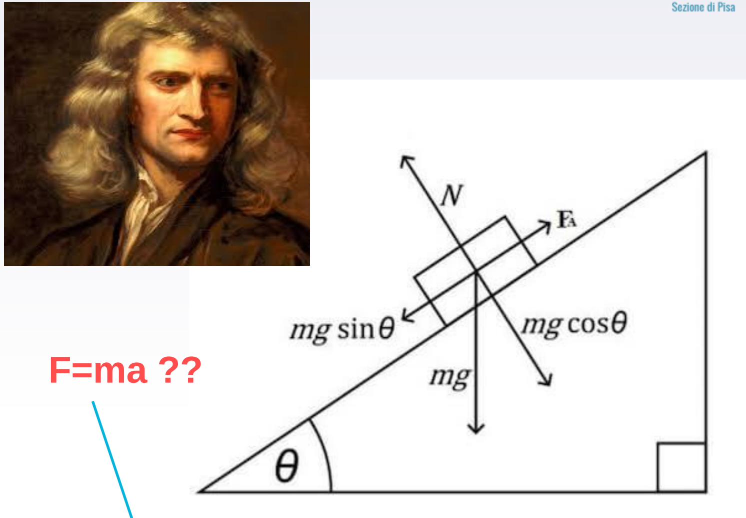
- Apart from some limits on the electronics *(“I cannot dispatch more than X consecutive triggers”*), **the real limit** on the numbers and type of events collected by HEP experiments **is the Computing**, and on its turn the **amount of money** one can dedicate to that.
- If you want, it is a reversed 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 experiment**

Ok for the storage, but CPUs?

- Up to here, we discussed the storage needs; **it turns out that CPU power is also a problem**
- Where do we spend CPU time in current HEP experiments?
- Broad brush list – se later for details
  - Interpretation of RAW detector signals into physics objects (“**Reconstruction**”)
  - Statistical studies of the reconstructed events (“**Analysis**”)
  - Simulation of the physics processes (“**Generators**”), the detector response (“**Simulation**”), the electronics (“**Digitization**)”

Why simulation?

- The largest part of our activities is comparing hypotheses with the data we collect
- 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…



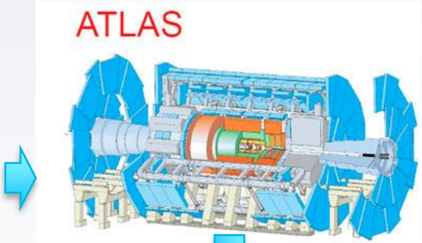
F=ma ??

$$t = \sqrt{\frac{2l}{g \,\mathrm{sen}\,\theta}}$$

Reality

LHC collisions

Decay of unstable particles

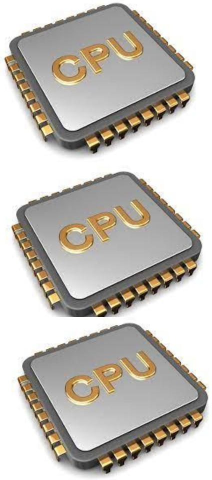


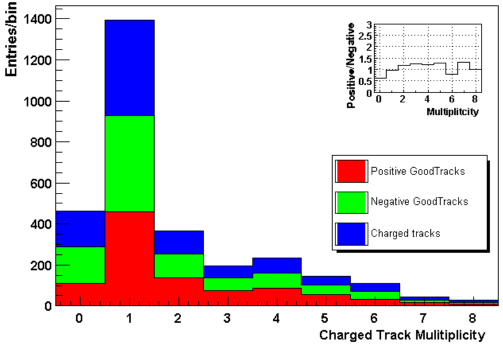
Detector electronics

Trigger

Reconstruction

Analysis





Simulation

Theoretical model

Simulation of decays of unstable particles

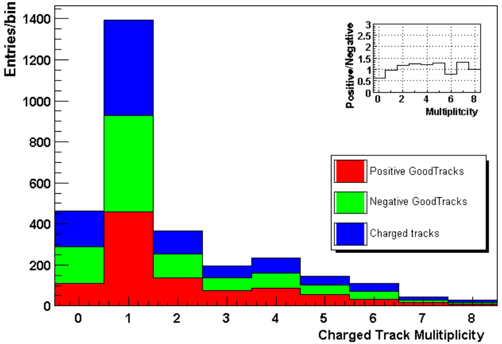
Simulation of interactions particle-detector

Simulation of detector electronics

Trigger

Reconstruction

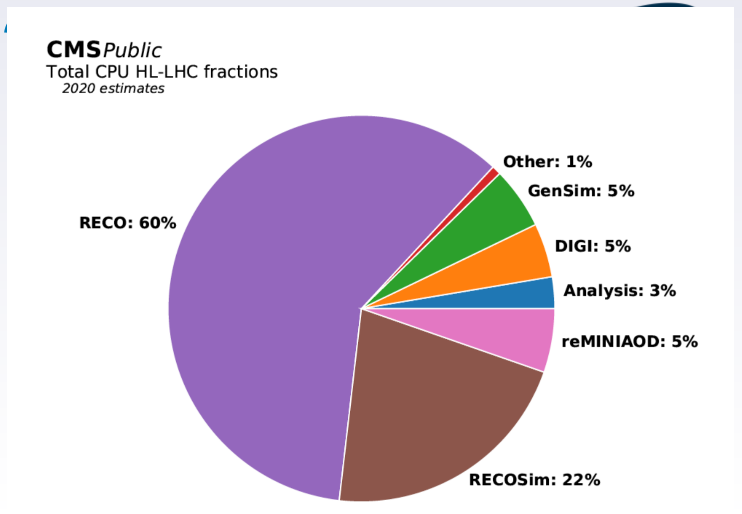
Analysis



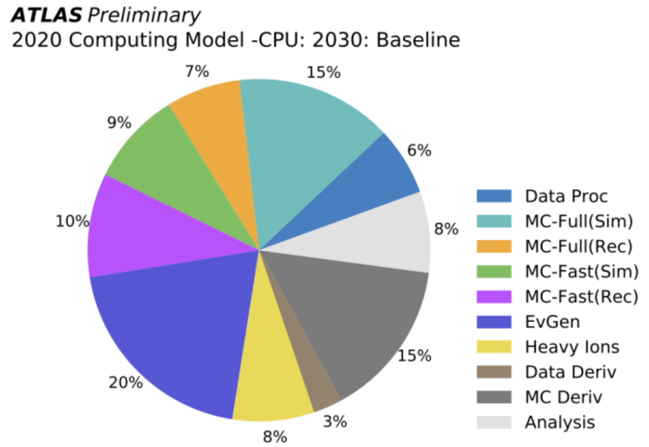
Where do we spend CPU

- 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



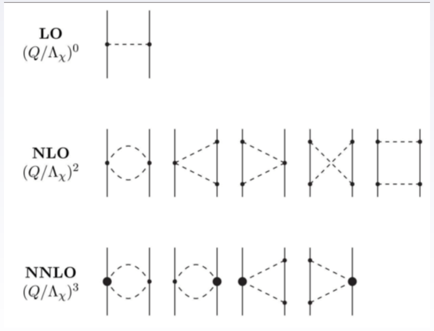
CMS *Public*
Total CPU HL-LHC fractions
2020 estimates

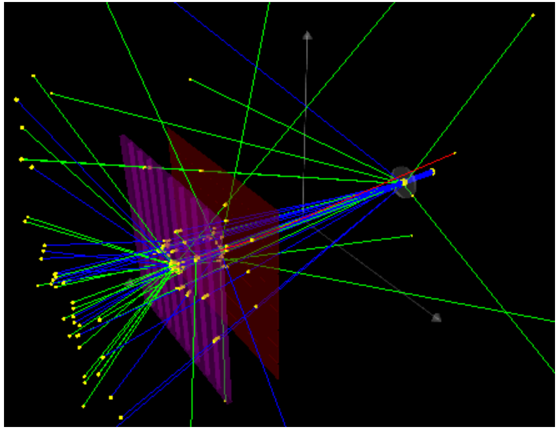


ATLAS *Preliminary*
2020 Computing Model -CPU: 2030: Baseline

towards absolute numbers

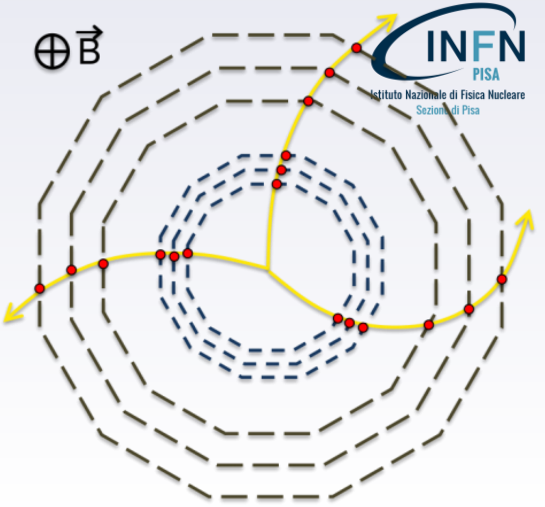
- **Event Generation**: depends strongly on the generator choses (Madgraph vs Sherpa vs PowHeg vs …) and the precision requested (LO vs LNO vs NLO vs …)
- **Simulation**: by now, the vast majority (all?) the experiments use **Geant4** as the simulation toolkit; still, its requested resources depend on stuff like: *volume of the detector, number of volumes, intrinsic detector resolution, importance of low energy secondary interactions,*

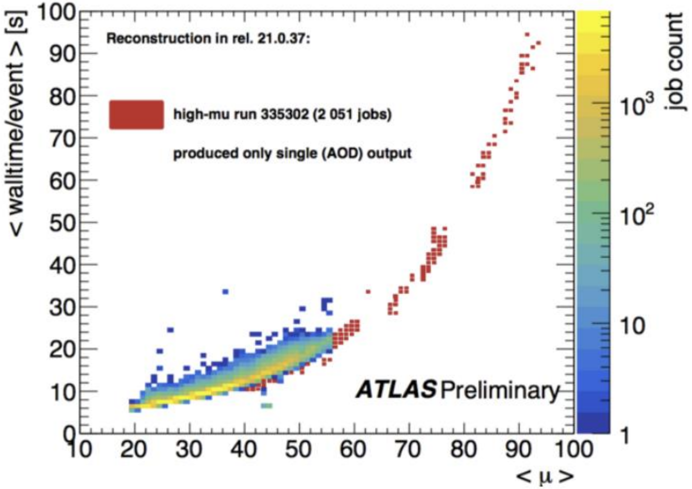


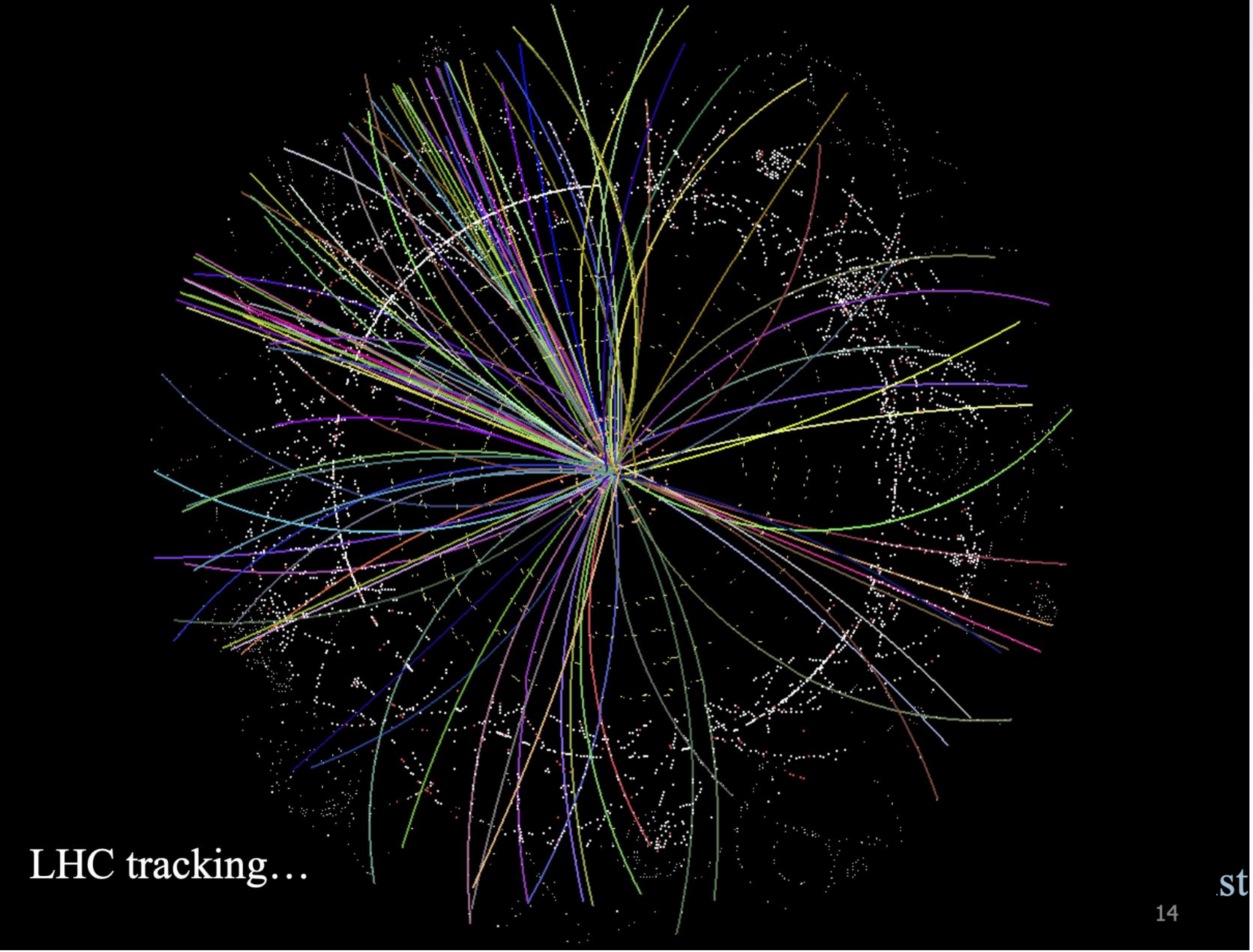


...

- **Analysis** the task which tries to interpret the signals from the (simulated) detector is terms of quantities interesting to the particle physicist (leptons, jets, vertices, …)
- The most time consuming task is “tracking reconstruction” using very high res detectors (typically thin silicon layers). It is a good example since
  - It is mathematically complex (Kalman Filter, matrix algebra, propagation in a not uniform magnetic field)
  - It is highly combinatorial: given a set of N signals, it scales as N^M, with M>1 and depending on your algorithm
- This is typical today → see later for how Machine Learning is changing the picture







LHC tracking…

st

...

- **Analysis** is mostly selection of events, with statistical interpretation
  - Selection can mean running ad-hoc reconstruction steps, hence not CPU ch
  - Statistical interpretation is today a quite CPU intensive activity:
    - high dimension likelihoods on millior billions of events
    - utilization of Toy Monte Carlos to correctly estimate correlated errors

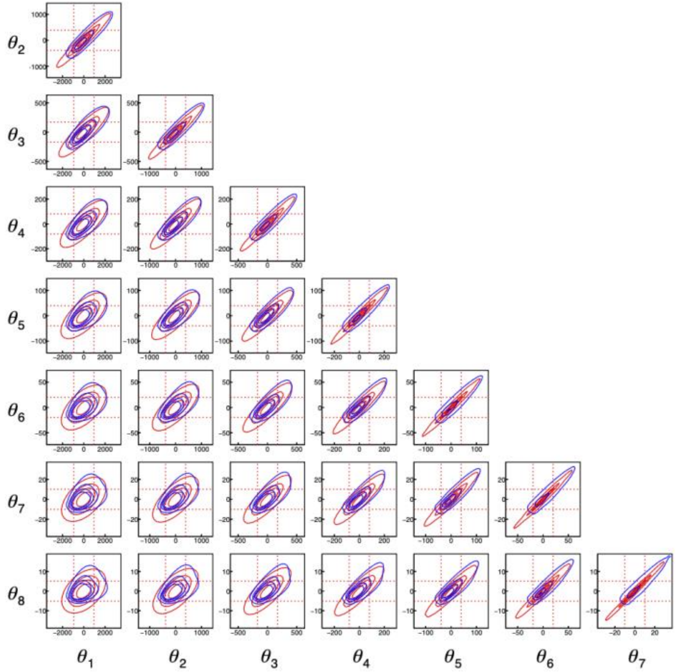


Figure 9: Contours of constant probability density for the true probability density function and the Gaussian approximation for the nuisance parameters in the toy search where an asymmetric background systematic is included. The red dotted horizontal and vertical lines indicate the regions for which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

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 is being used since 2009: HepSpec06 (HS06)
  - **Rule of thumb**: a CPU "core" today is ~10-20 HS06
  - Hence, a 64 core CPU is ~ 1000 HS06
  - **Hence, a 2 CPU box is today ~ 2000 HS06. Since it costs ~7000 CHF, the current price estimate is ~ 3.5 CHF/HS06**

Absolute numbers …

- Today, with standard Run-2 LHC, typical numbers in CMS/ATLAS are
  - **Event generation**: 100-1000 HS06s per event (which means ~ 10-100 sev/ev on a single Xeon core)
  - **Simulation (G4):** 500-3000 HS06s
  - **Reconstruction**: 150-300 HS06s
  - **Analysis**: can be anything, usually quite fast (<1-100 HS-06s)

- With these numbers, we can try and project the Computing (CPU and storage) needs for a HEP experiment today,

So 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 ev*300 sec*HS06/ev = 2e12 sec*HS06 = 70000 HS06 for the entire year (→ 7000

MC
2x110002x70000 HS06 reconstruction
0 HS06 simulation
Analisys (MC + DT):
7e9ev*2*10 sec*HS06/sec *N = 1.4e11 sec*HS06 *N = 4500*N HS06
Where N is the number of independent analyses,can be very high (~100)

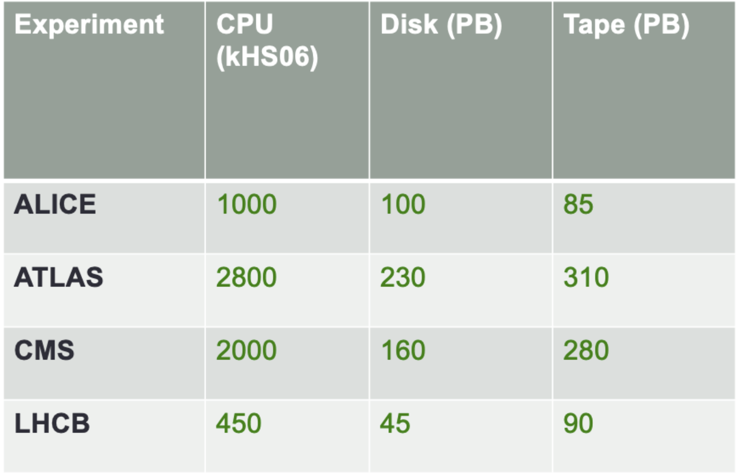
TOTAL: 70000+140000+220000+450000 ~ 1M HS06

Today they are
3000 HDD/y
100000 computing cores

.. And these are per experiment for a single year of data taking!

Reality is higher …

- 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 one
  - Analyzing data from previous + current year
  - Taking data in the current year
  - Preparing future data taking periods an detector upgrades
- So, all in all, real resource number per experiment are **underestimated by at leas factor 3x**

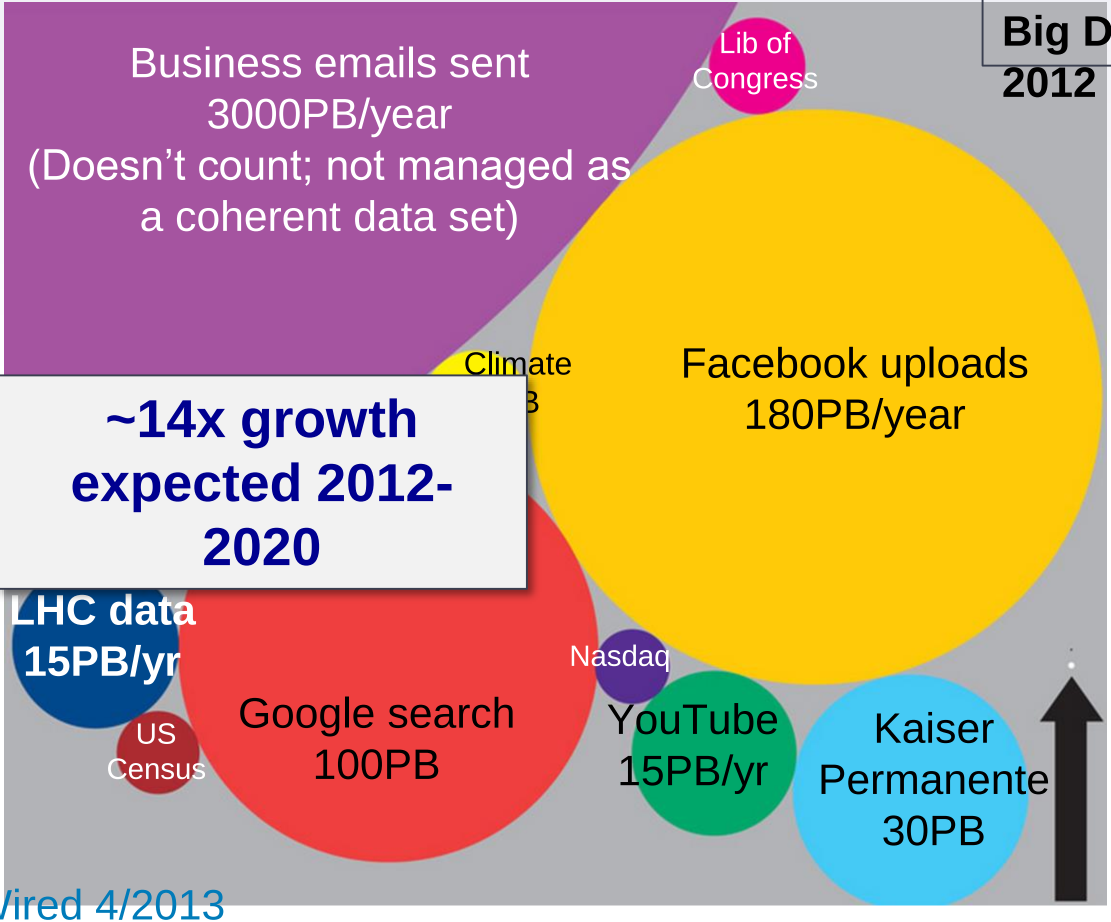


Experiment	CPU (kHS06)	Disk (PB)	Tape (PB)
ALICE	1000	100	85
ATLAS	2800	230	310
CMS	2000	160	280
LHCB	450	45	90

How to handle this?

- By today’s metric, handling ~ 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 (today ~ 1000x), and a CPU was probably 0.1 HS06 (today ~ 10000x)
- 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 wo

Comparison with the rest of the world - 2012



Big Data in 2012

~14x growth expected 2012-2020

We are big…
not NSA-big, but big
(and more cost efficient)

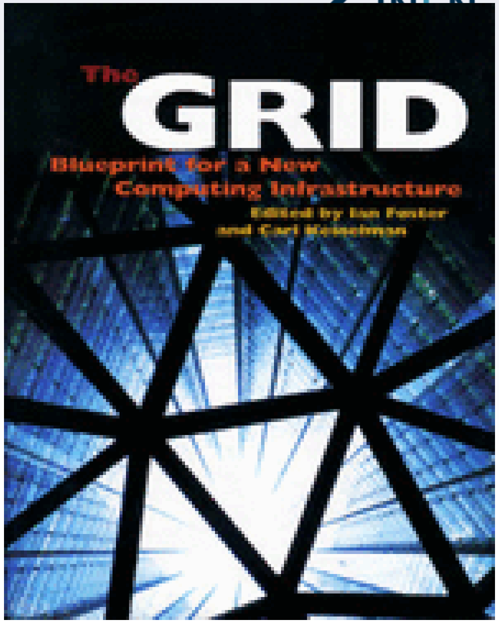
Current ATLAS data set, all data products: 140 PB

Wired 4/2013

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

1. **Build a BIG data center**
    1. 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…
    2. 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, ...
2. **Many small data centers**
    1. 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

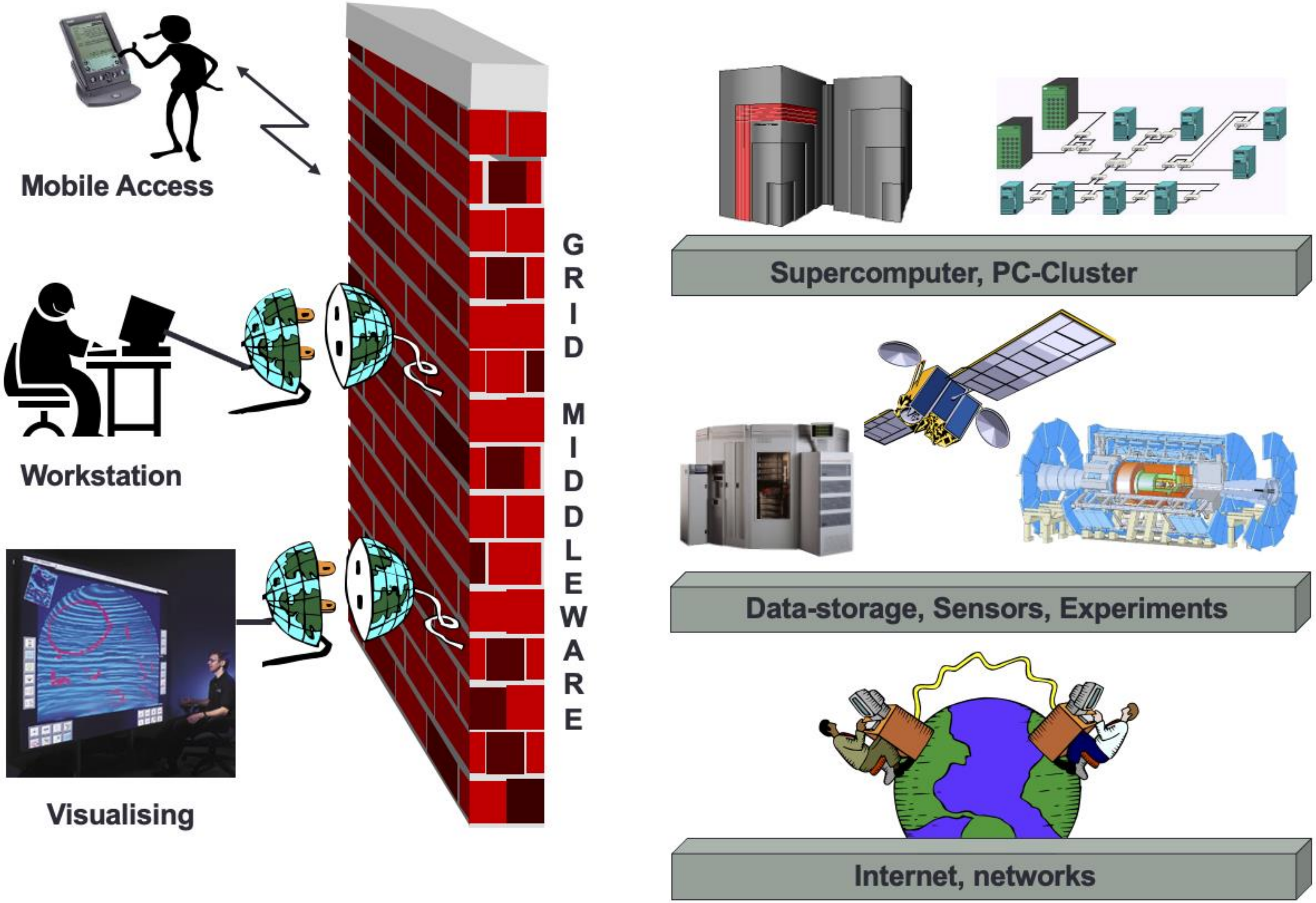
“When the network is as fast as the computer's internal links, the machine disintegrates across the net into a set of special purpose appliances”

(George Gilder)

The idea in a nutshell

Split the problem into two levels:

- **The physical level:**
  - Distribute resources worldwide in N (>100) centers
  - Technically is 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

INFN PISA
Istituto Nazionale di Fisica Nucleare
Sezione di Pisa

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

GGUS

INDIGO - DataCloud
iam

X.509

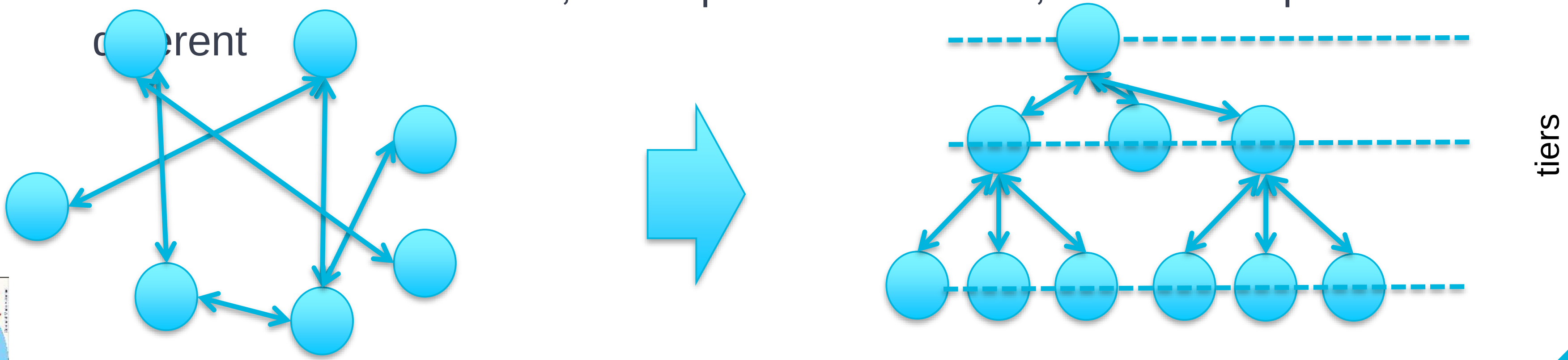
RUCIO

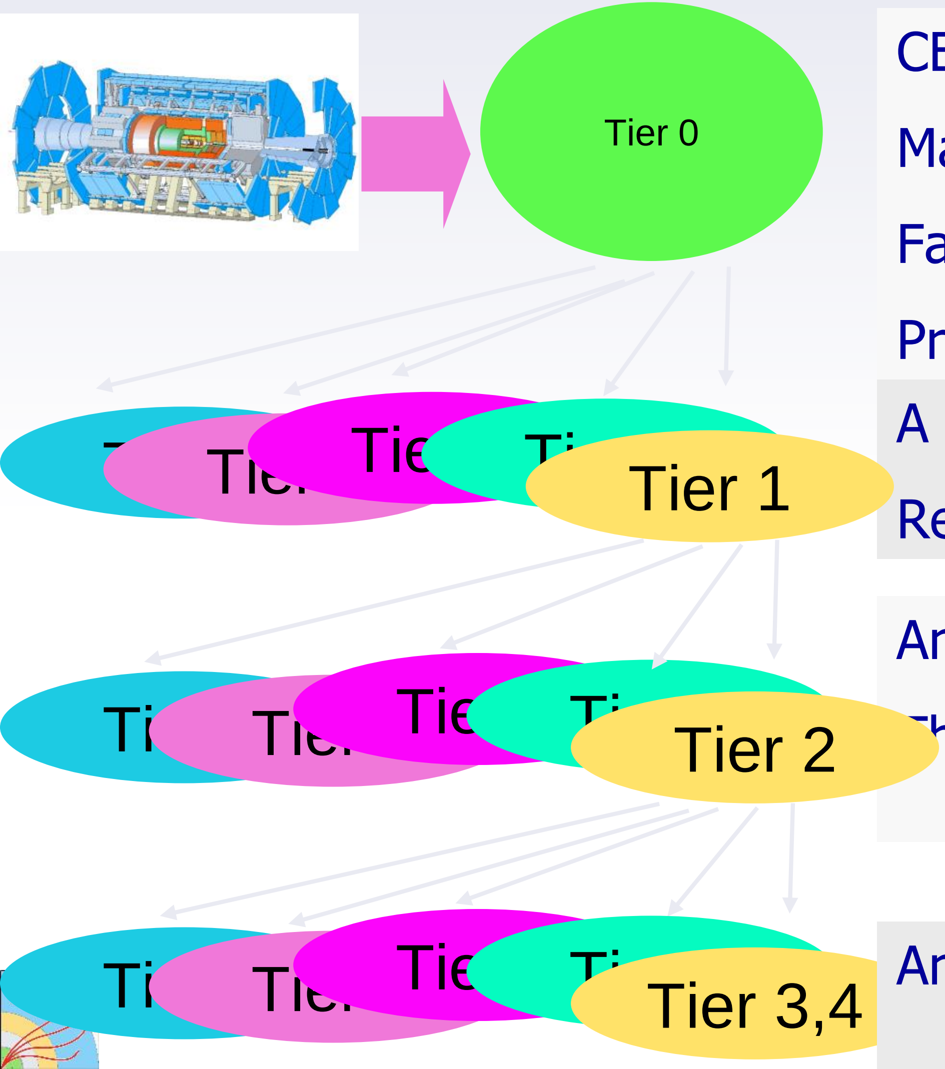
Map
Satellite
Terrain
ASIA
Atlantic Ocean
AFRICA
SOUTH AMERICA
Pacific Ocean
Indian Ocean
AUSTRALIA
Pacific Ocean
POWERED BY Google
5000 mi
5000 km
Imagery ©2012 NASA - Terms of Use

CMS

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 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 → 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





CERN

Master copy of RAW data

Fast calibrations

Prompt Reconstruction

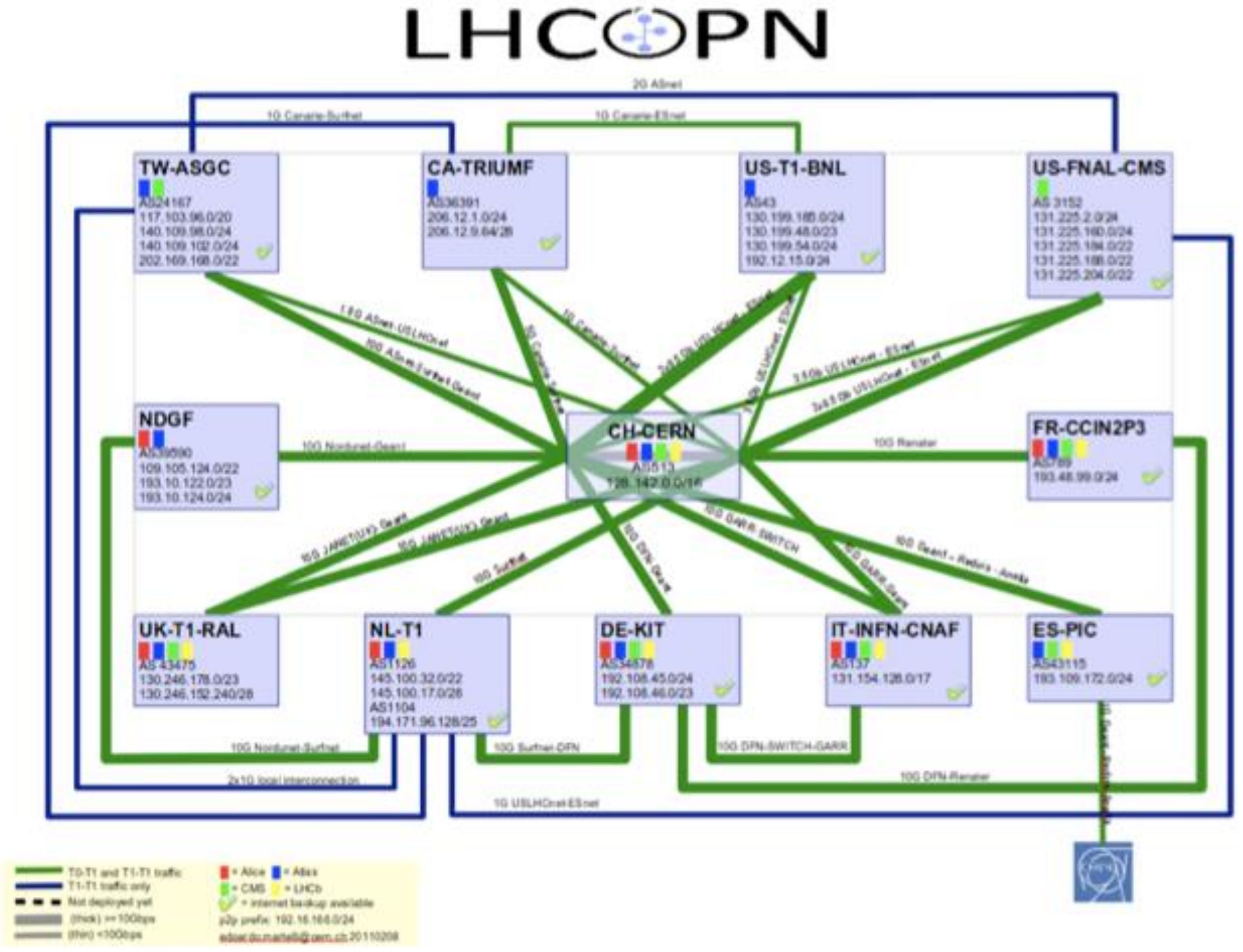
A second copy of RAW data (Backup)

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



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

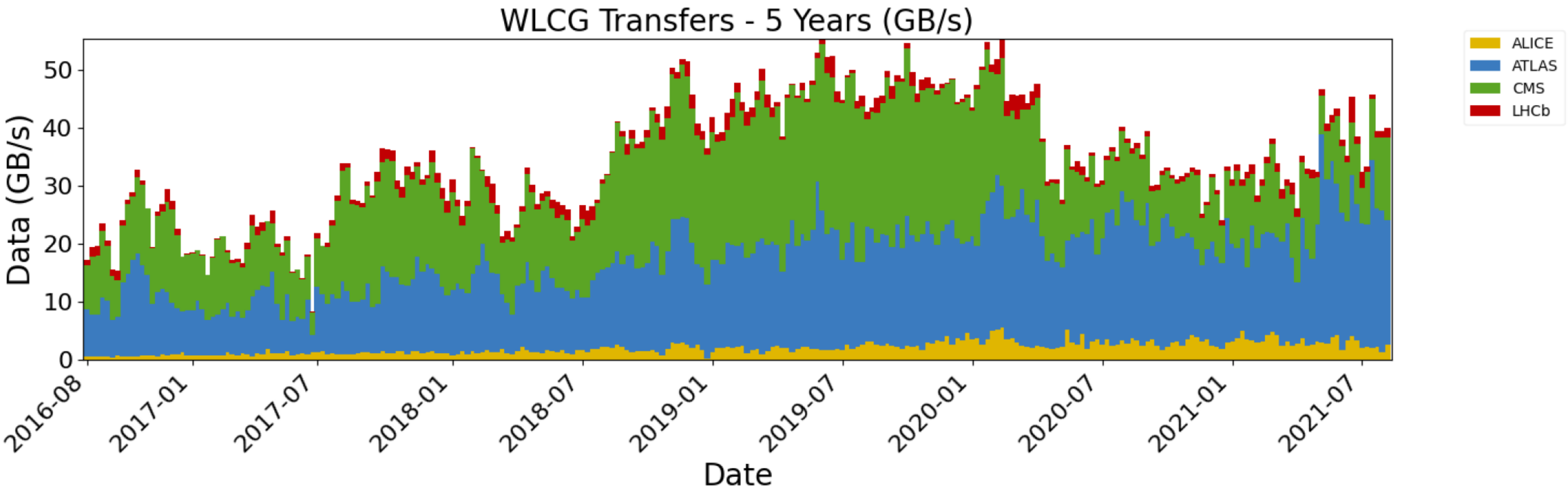
1st copy stays @ CERN, but a 2nd copy must go distributed for disaster recovery

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

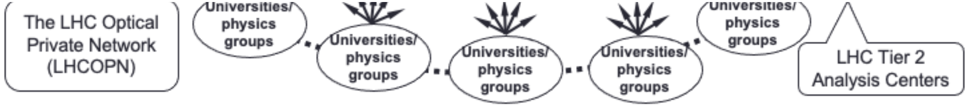
→ **By today , multiple of 100 Gbps**

A Network Centric View of the LHC

CERN →T1	miles	kms
France	350	565



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

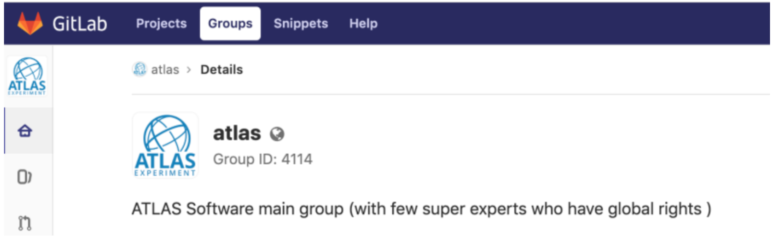

The LHC Optical Private Network (LHCOPN)
Universities/ physics groups
Universities/ physics groups
Universities/ physics groups
Universities/ physics groups
Universities/ physics groups
LHC Tier 2 Analysis Centers

The software!

- How big / how complex?
- The 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

Let’s use ATLAS and CMS as examples

- Code published with Apache 2.0 license (“free”)
  - https://gitlab.cern.ch/atlas
  - https://github.com/cms-sw/cmssw

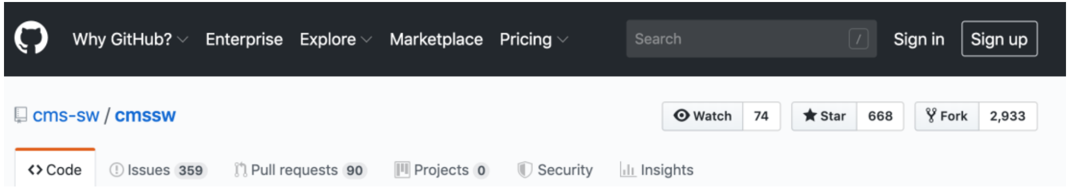


atlas › Details

atlas

Group ID: 4114

ATLAS Software main group (with few super experts who have global rights)



cms-sw / cmssw

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

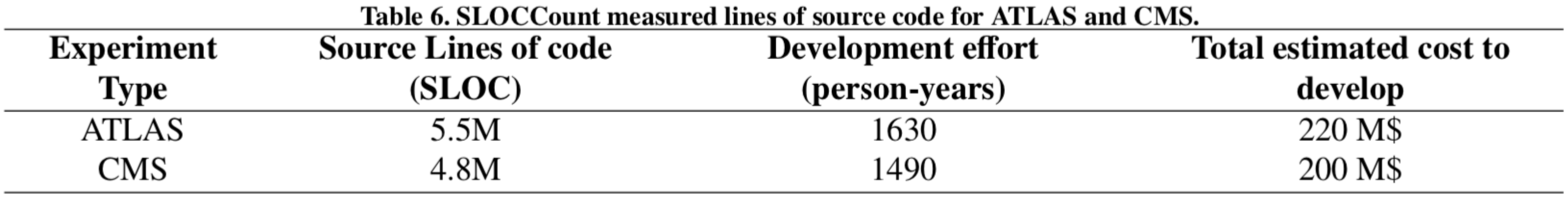


Table 6. SLOCCount measured lines of source code for ATLAS and CMS.

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-ccompiler 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-cxxcompi gcc-atomic gcc-checker-plugin gcc-plugin gdb geant4-parfullcms geant4data py2-numpy openloops git glibc glimpse gmake gnuplot gosam gosamcontrib hdf5 igprof intel-license ittnotify 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-functooccj 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-gitdb2 py2-gitpython py2-google-common py2-googlepackages py2-grpcio py2-h5py py2-h5py-cache py2-hep_ml py2-histbook py2-histogrammar py2-html5lib py2-hyperas py2-hyperopt py2-idna py2-ipaddress py2-ipykernel py2-ipython py2-ipython_genutils py2-ipywidgets py2-jedi py2-jinja2 py2-jsonpickle py2-jsonschema py2-jupyter py2-jupyter_client py2-jupyter_console py2-jupyter_core py2-keras py2-keras-application py2-keras-preprocessi 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-neurolab 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-parso 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-pygithub py2-pygments py2-pymongo py2-pyopenssl py2-pyparsing py2-pysqlite py2-pytest py2-python-cjson py2-python-dateutil py2-python-ldap py2-pytz py2-pyyaml py2-pyzmq py2-qtconsole py2-rep py2-repoze-lru py2-requests py2-root_numpy py2-root_pandas py2-rootpy py2-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-theanets py2-theano py2-thriftpy py2-tornado py2-tqdm py2-traitlets 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-widgetsnbextensio py2-xgboost py2-xrootdpyfs pydata pyminuit2 pyqt python-paths python_tools rootglew scons sloccount tcmalloc tcmalloc_minimal tensopy2-virtualenvwrapperflow tinyxml2 xtl blackhat boost boost_header python bz2lib cascade_headers ccache-ccompiler 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 hepmc_headers distcc-ccompiler distcc-cxxcompiler distcc-f77compiler dpm expat fastjet fftjet fftw3 freetype gbl gdbm gsl giflib google-benchmark libjpeg-turbo hector heppdt **madgraph5amcatnlo** llvm-cxxcompiler jemalloc jimmy_headers ktjet libhepml libuuid **llvm-ccompiler** llvm-f77compiler meschach mxnet-predict numpy-c-api x11 oracle pacparser yoda protobuf **python3** qd_f_main sqlite sigcpp tauola_headers **tbb** **tensorflow-framework** tensorflow-runtime tensorflow-xla_compil0-pafccj3 toprex_headers utm valgrind vdt_headers xrootd xtensor boost_system 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-cusparse cuda-npp cuda-nvgraph cuda-nvjpeg cuda-nvml cuda-nvrtc das_client vecgeom hepmc frontier_client google-benchmark-main libpng iwyu-cxxcompiler libtiff libungif llvm-analyzer-ccompil llvm-analyzer-cxxcomp mcdb opengl openldap oracleocci pyclang qtbase sip starlight tauola tensorflow-c tensorflow-cc tkonlinesw toprex vdt boost_chrono boost_filesystem boost_mpi cgal lhapdf classlib davix rootcling **geant4core** photospp geant4static graphviz lwtnn millepede qt3support rivet tkonlineswdb cgalimageio herwig rootmathcore rootrio **pythia8** geant4vis thepeg pyquen qt rootrint rootrflx rootsmatrix rootx11 sherpa charybdis rootthread dire tauolapp **geant4** geneva herwigpp jimmy qtdesigner rootgeom rootxmlio vincia rootcore evtgen roothistmatrix rootmath rootxml rootphysics rootgpad rootfoam rootspectrum root rootminuit rootgraphics rootgui rootinteractive roothtml rootminuit2 dd4hep-core roofitcore mctester professor2 rooteg rootgeompainter rootrgl rootged rootguihtml rootmlp rootpy **dd4hep** dd4hep-geant4 roofit rooteve roottmva roostats 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, and not often done during data taking (CDF and Babar can be exceptions). The CMS case:
    - **Y< 2000**: CMSIM+CMKIN (Fortran + Geant3)
    - **2000<Y<2005**: OSCAR + ORCA (C++ + Geant4 + ObjectivityDB/ROOT)
    - **Y>2005**: CMSSW (C++ + Geant4 + ROOT + Python)
    - The last «change» (ORCA to CMSSW) took from 2004 to 2007 to reach the same level, with 2 devel teams needed (the old SW used for a TDR while preparing the new one)
    - **Data taking started in 2008**

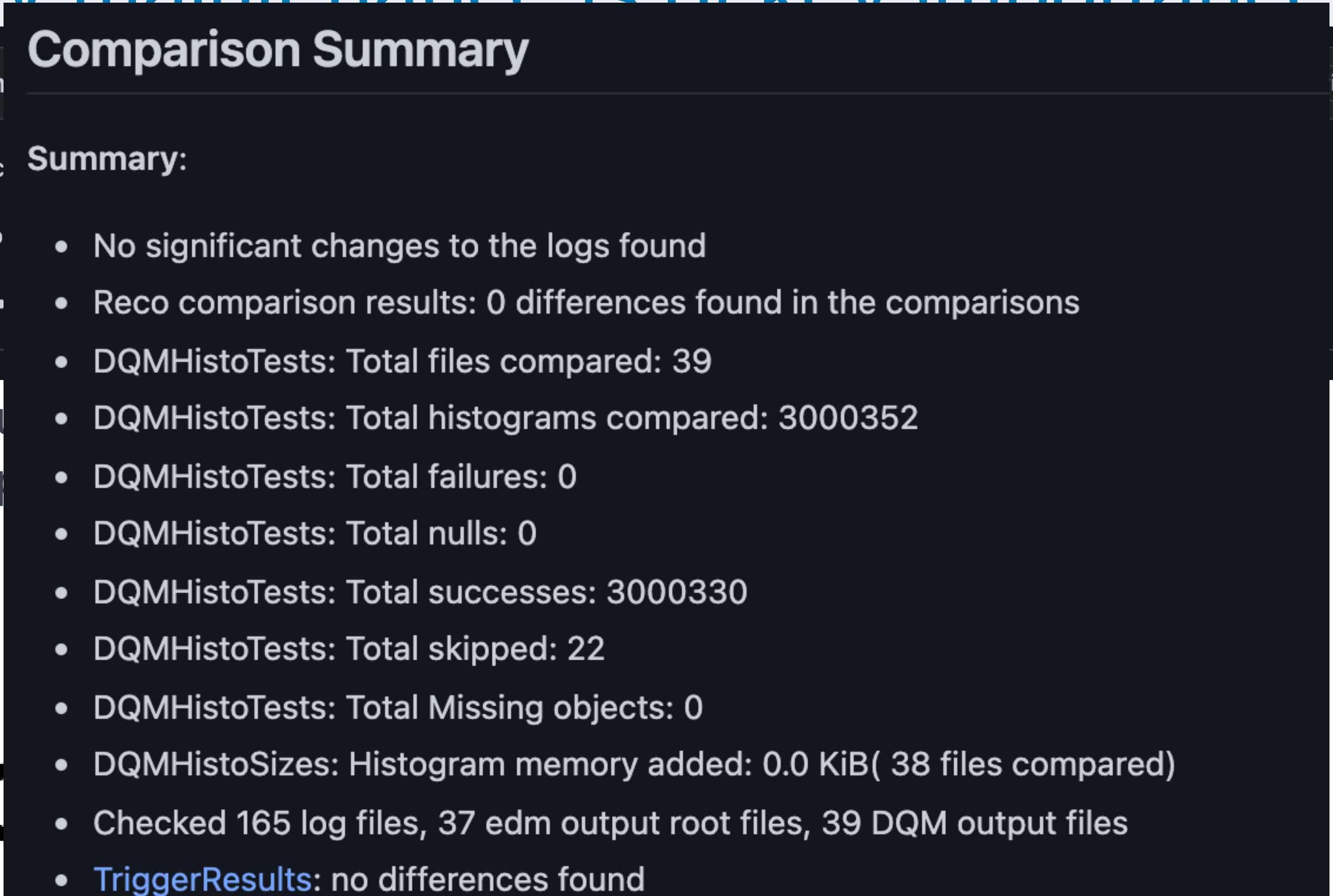
Typical needs from a framework …

- **Modularity**: large utilization of plugins to late-bind algorithms, pieces of code, 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 (COW) to multi threaded (TBB)
From single core PCs to O(300) cores per PC (KNL)
From configs to Python as the uber language
From fully scheduled execution to unscheduled (needed for multi threading)
Analysis support from PAW-ROOT(cint)-ROOT(cling)-PyROOT-UpROOT-AVRO

SW maintenance is of key importance



Comparison Summary

Summary:

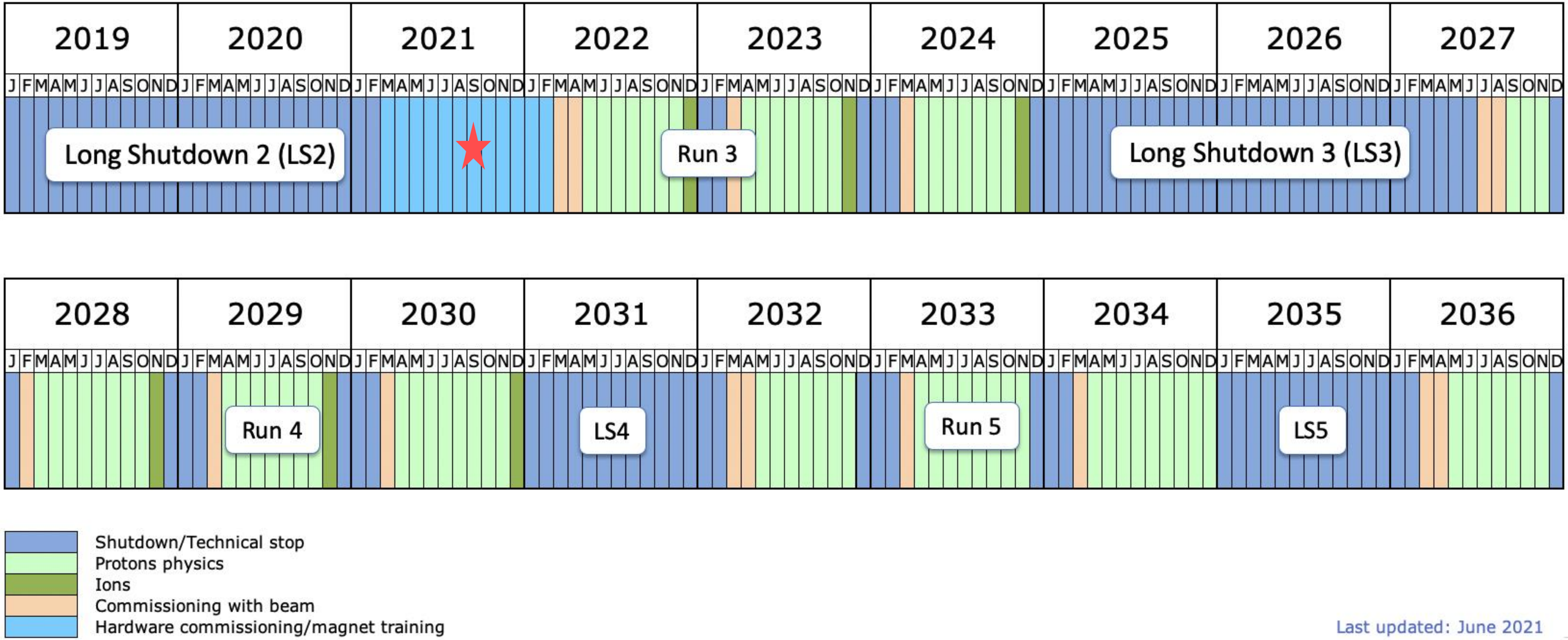
- No significant changes to the logs found
- Reco comparison results: 0 differences found in the comparisons
- DQMHistoTests: Total files compared: 39
- DQMHistoTests: Total histograms compared: 3000352
- DQMHistoTests: Total failures: 0
- DQMHistoTests: Total nulls: 0
- DQMHistoTests: Total successes: 3000330
- DQMHistoTests: Total skipped: 22
- DQMHistoTests: Total Missing objects: 0
- DQMHistoSizes: Histogram memory added: 0.0 KiB(38 files compared)
- Checked 165 log files, 37 edm output root files, 39 DQM output files
- TriggerResults: no differences found

The future ….

- **“it all works”, so why change?**
- We have the proof that the computing systems for today’s collider experiment do work. CMS and ATLAS have published > 1000 papers each, ALICE and LHCb ~ 500
- Computing is a large operational cost; but is ~ constant year over year and somehow possible to cover ….
- **Are we done? No we are not …**

The medium term future for HEP

- HL-LHC:

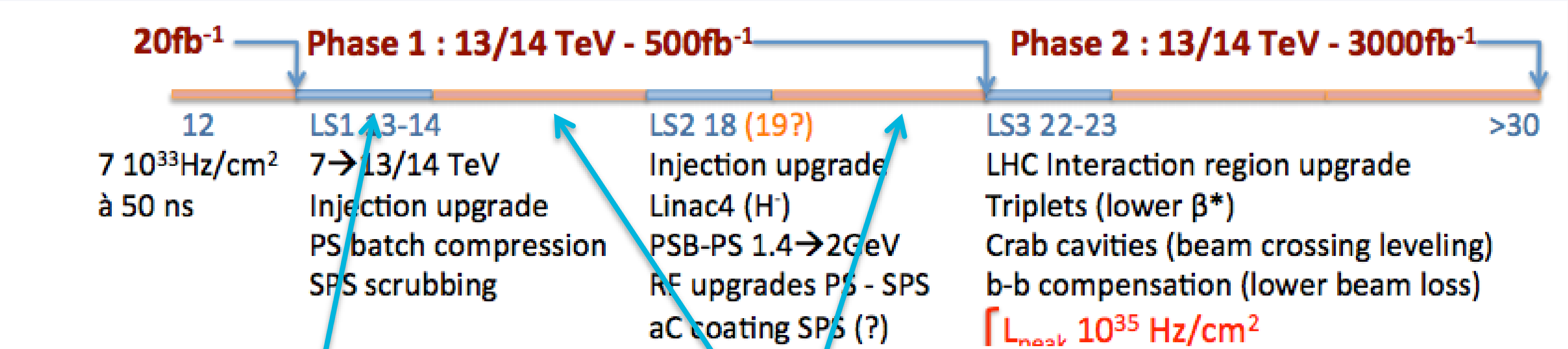


Last updated: June 2021

🟦 shutdown

🟧 run

Run 1 Run 2 Run 3 Unknown territory…



$20fb^{-1}$ → Phase 1 : 13/14 TeV - $500fb^{-1}$ → Phase 2 : 13/14 TeV - $3000fb^{-1}$

12
$7\ 10^{33}Hz/cm^2$
à 50 ns

LS1 13-14
7→13/14 TeV
Injection upgrade
PS batch compression
SPS scrubbing

LS2 18 (19?)
Injection upgrade
Linac4 (H^-)
PSB-PS 1.4→2GeV
RF upgrades PS - SPS
aC coating SPS (?)

LS3 22-23
LHC Interaction region upgrade
Triplets (lower β*)
Crab cavities (beam crossing leveling)
b-b compensation (lower beam loss)
[L_{peak} 10^{35} Hz/cm^2

>30

- 2015-2018: 13 TeV, ~2.5x in luminosity, up to 3x in hadronic events per collision
- 2021-2023: 13 TeV, again 2.5x in luminosity
- 2026+: the so-called HL-LHC (or SLHC)
- 2035+: the so-called HE-LHC @ 30 TeV
  - Magnet for this simply do not exist at them moment, so wait and see
- (2040+: at some point I will retire, so no more my problem…)

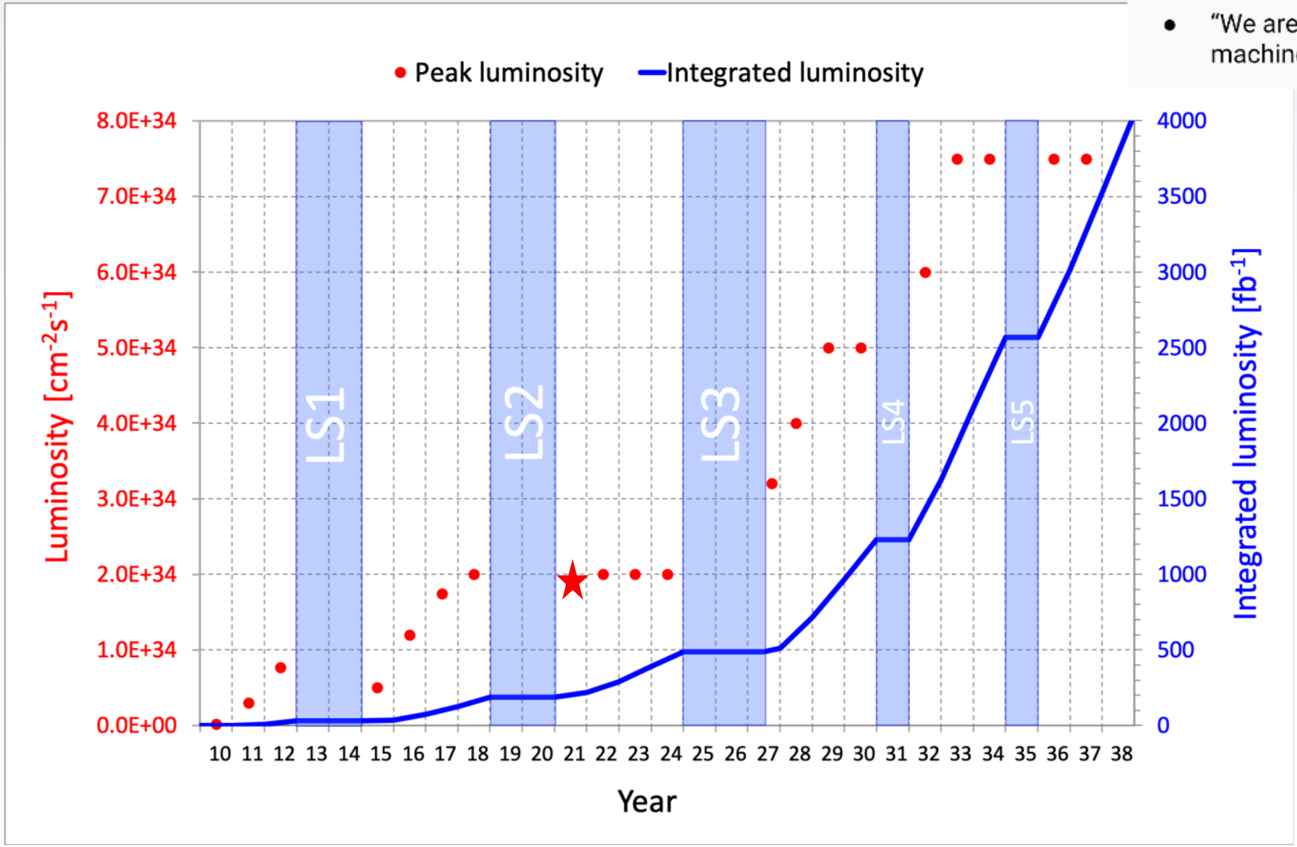
We are here

HL-LHC

Some true but amazing statements:

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

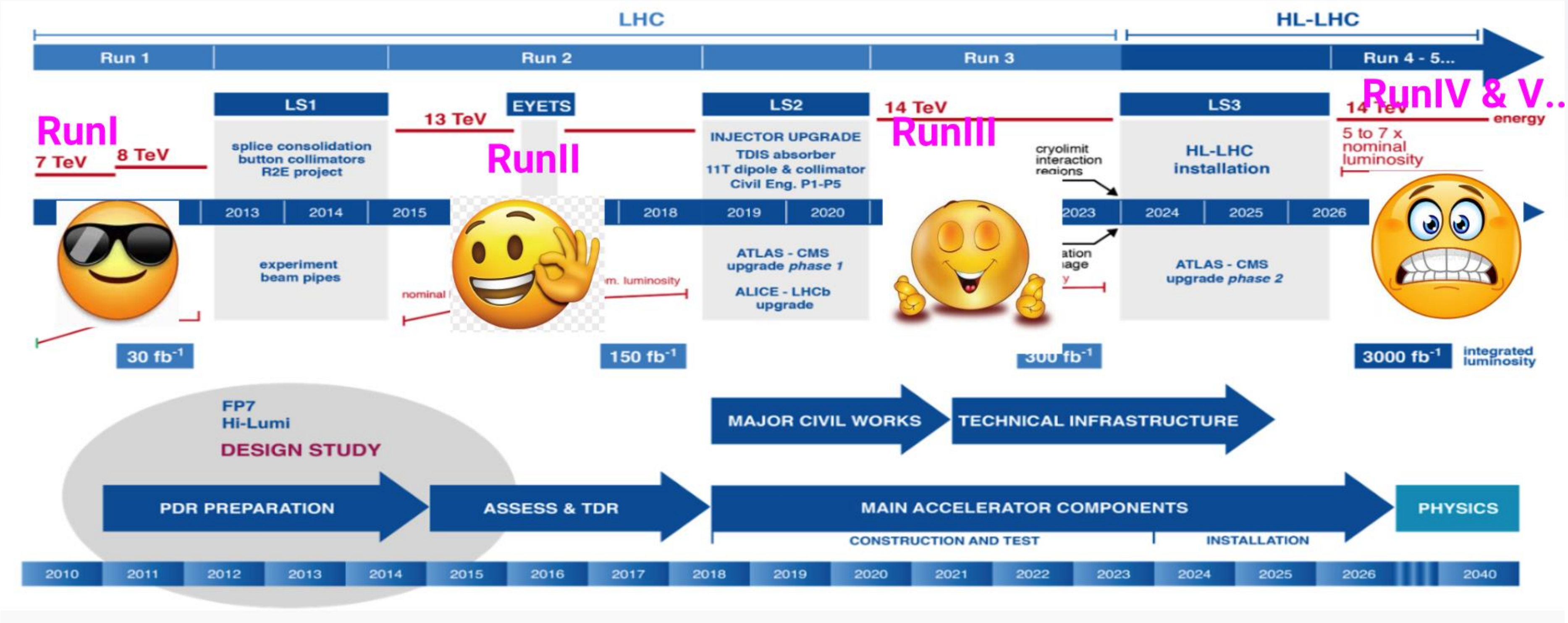
~proportional
to the pile-up and hence
to single event complexity



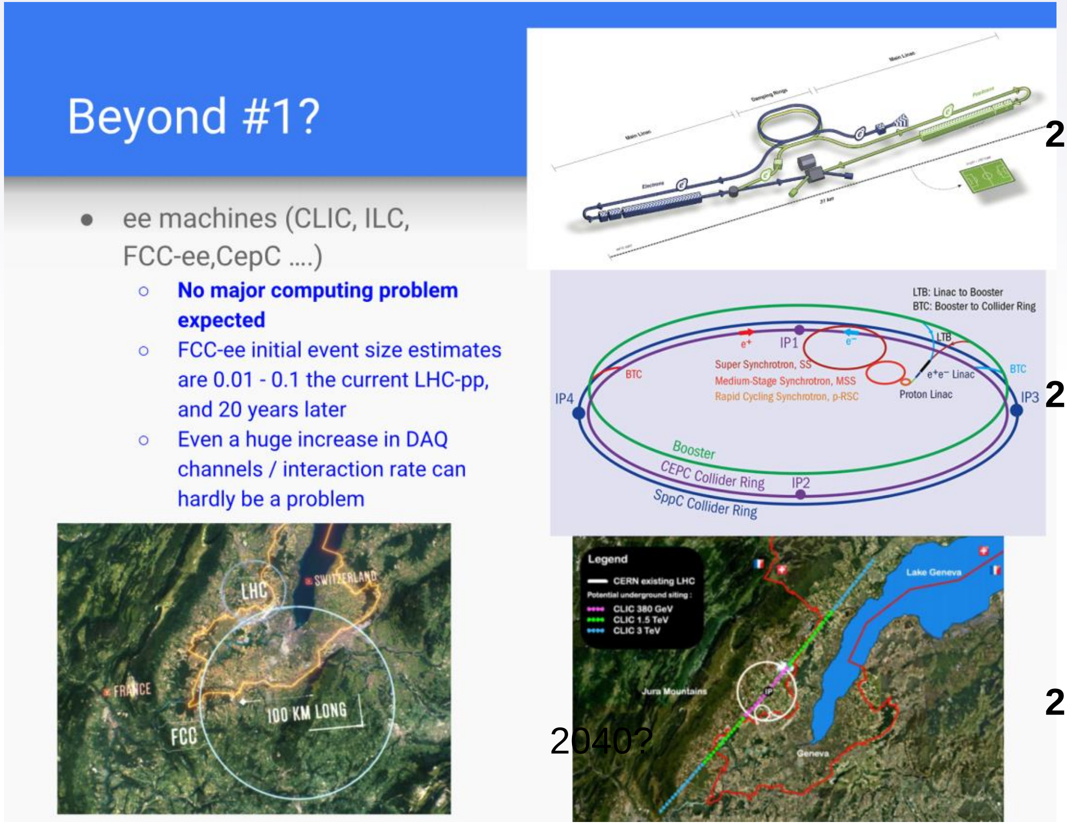
~proportional
to the total number
of generated
collisions

We are here!

And for computing?



HL-LHC is not the end of the story!



Beyond #1?

- ee machines (CLIC, ILC, FCC-ee,CepC ….)
    - **No major computing problem expected**
    - FCC-ee initial event size estimates are 0.01 - 0.1 the current LHC-pp, and 20 years later
    - Even a huge increase in DAQ channels / interaction rate can hardly be a problem

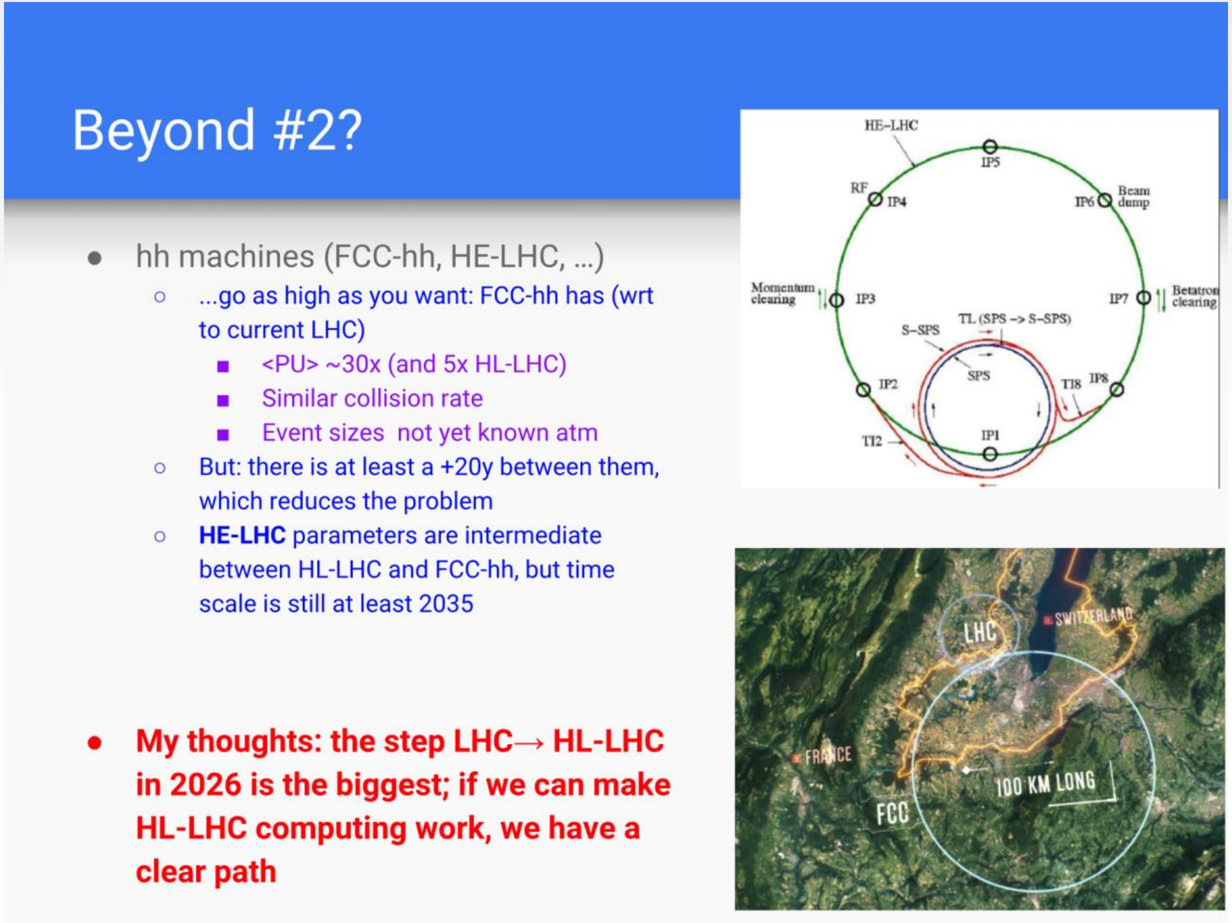
Main Linac
Damping Rings
Main Linac
Positrons
Electrons
31 km
LTB: Linac to Booster
BTC: Booster to Collider Ring
IP1
e+
e−
LTB
BTC
BTC
Super Synchrotron, SS
Medium-Stage Synchrotron, MSS
Rapid Cycling Synchrotron, p-RSC
e+e− Linac
Proton Linac
IP4
IP3
Booster
CEPC Collider Ring
IP2
SppC Collider Ring
LHC
SWITZERLAND
FRANCE
FCC
100 KM LONG
Legend
CERN existing LHC
Potential underground siting :
CLIC 380 GeV
CLIC 1.5 TeV
CLIC 3 TeV
Lake Geneva
Jura Mountains
IP
Geneva

2040?

2035?

2040?

2040?



Beyond #2?

- hh machines (FCC-hh, HE-LHC, ...)
  - ...go as high as you want: FCC-hh has (wrt to current LHC)
    - <PU> ~30x (and 5x HL-LHC)
    - Similar collision rate
    - Event sizes not yet known atm
  - But: there is at least a +20y between them, which reduces the problem
  - **HE-LHC** parameters are intermediate between HL-LHC and FCC-hh, but time scale is still at least 2035

- **My thoughts: the step LHC→ HL-LHC in 2026 is the biggest; if we can make HL-LHC computing work, we have a clear path**

2040?

2045?

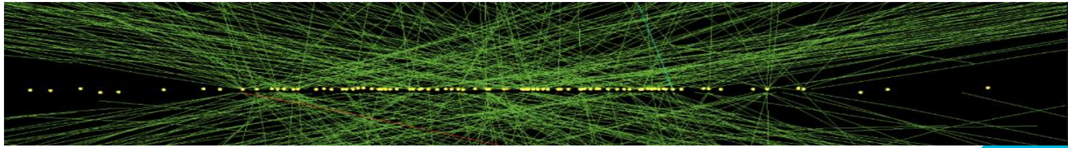
How are computing resources linked to machine / experiments parameters?

- # events collected = Experiment live time * Experiment rate to offline
  - LHC RunII: 7Msec * 1000 Hz = ~ 10 B events
- Bandwidth, total storage = # events collected * (1+ f_{MC}) * $F_{STORAGE}$(<PU>)
  - $F_{STORAGE}$(<PU>) ~ linear in <PU>
- Computing power = # events collected * (1 + f_{MC}) * F_{CPU}(<PU>)
  - F_{CPU}(<PU>) superlinear in <PU>
- Storage is also ~ integral with time
- $\text{Storage}_{\text{YearN+1}}$ = $\text{Storage}_{\text{YearN}}$ + $\text{Delta}_{\text{NEW EVENTS}}$

In the end, main parameters are

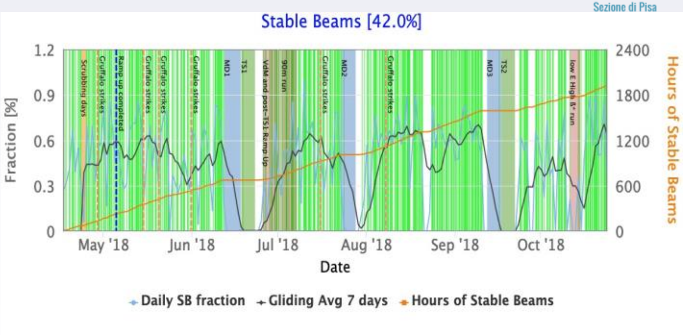
- Trigger rate
- Live time of the Accelerator
- <PU>
- f_{MC} (MC production needs)

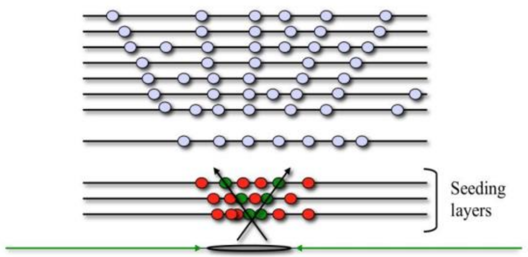
PU: the # of pp interactions per single bunch cossing



Scaling LHC → HL-LHC

- Main Evolution of important computing parameters
  - Live time cannot change much
  - <PU> goes from 35 to 200
  - Trigger rate 1 kHz → 7.5 kHz
- **HL-LHC / LHC = (7.5/1) * (200/35) = 42**
- This is optimistic!
  - Triggers have to remain clean
  - Assumes all is linear with <PU>, while reconstruction has at least a superlinear component
  - Upgraded detectors, more DAQ channels
- **A more realistic educated guess is 50-100x keeping all the rest constant**

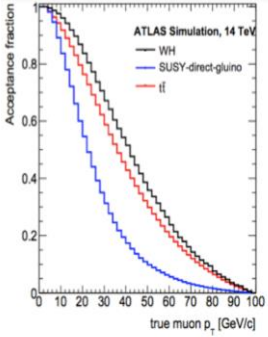




Trigger rate scales at best with $\mathcal{L}$ for

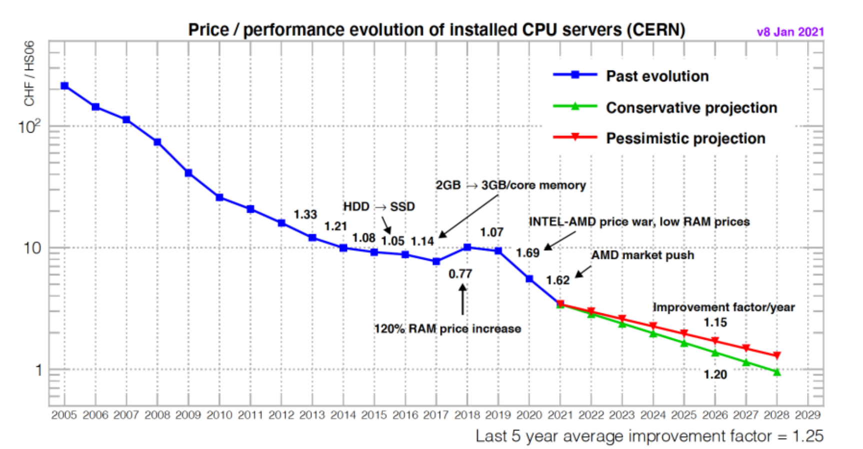
- Same physics
- Clean triggers

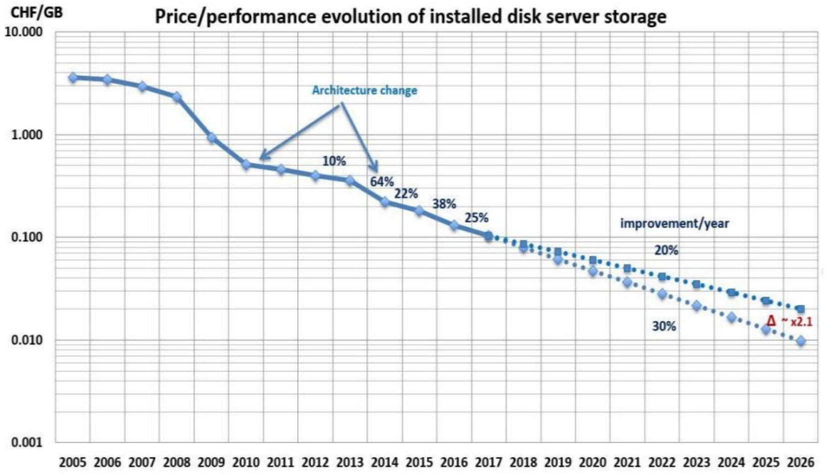
Difficult to do better than this

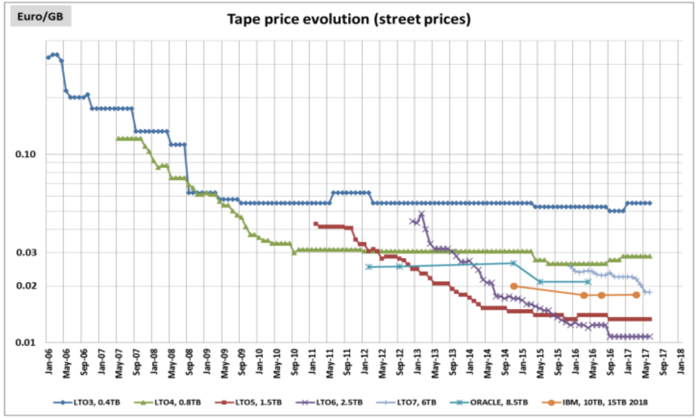


In the meantime, technology …

- Price per unit resource, from CERN procurement (B.Panzer)
- Moore’s (Kryder’s, Butter’s, Nielsen’s, …) law once predicted 2x/18 months (which is +60%/y)
- Now reasonable estimates are +15-20%/y
- **In the 7-8 years to HL-LHC, +20%/y is just 4x at fixed budget**
- **(50-100x)/4x = (12-25)x to “gain somewhere else”**







Question is …

Assuming we cannot get more money per year for computing, **where do we get the 12x-25x missing?**

A non final list

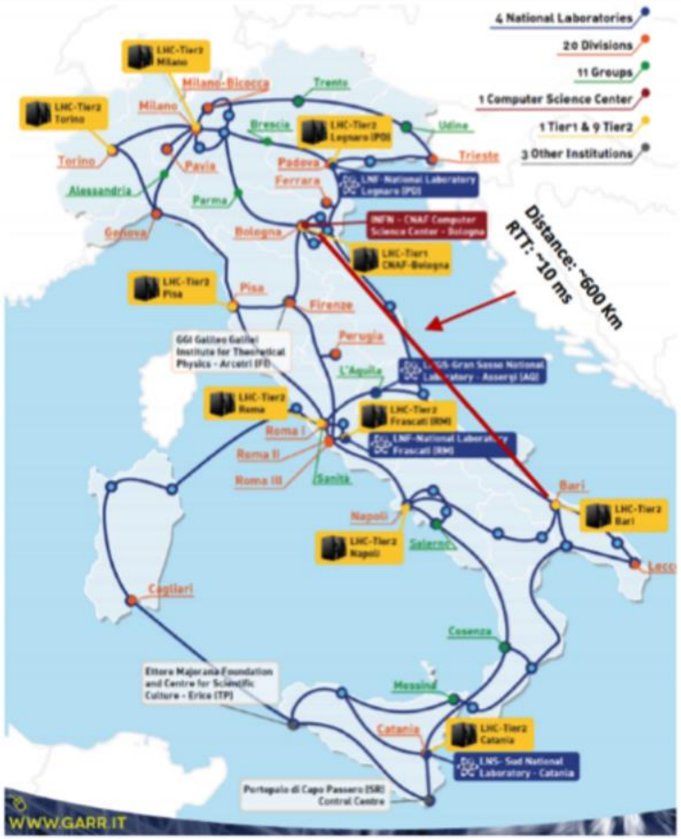
1. **Infrastructure** changes (where / how to get CPU and Disk, at which price)
2. **Technological** changes (use different technologies)
3. **Physics** #1: change analysis model (do the same physics with less resources)
4. **Physics** #2: reduce the physics reach (for example increasing trigger thresholds)
    - Not even considered here … it is the “desperation move” if we fail with everything else
5. Use “**modern weapons**” (new/faster algorithms/tools)
6. Something **unexpected**...

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 computing you can buy today?

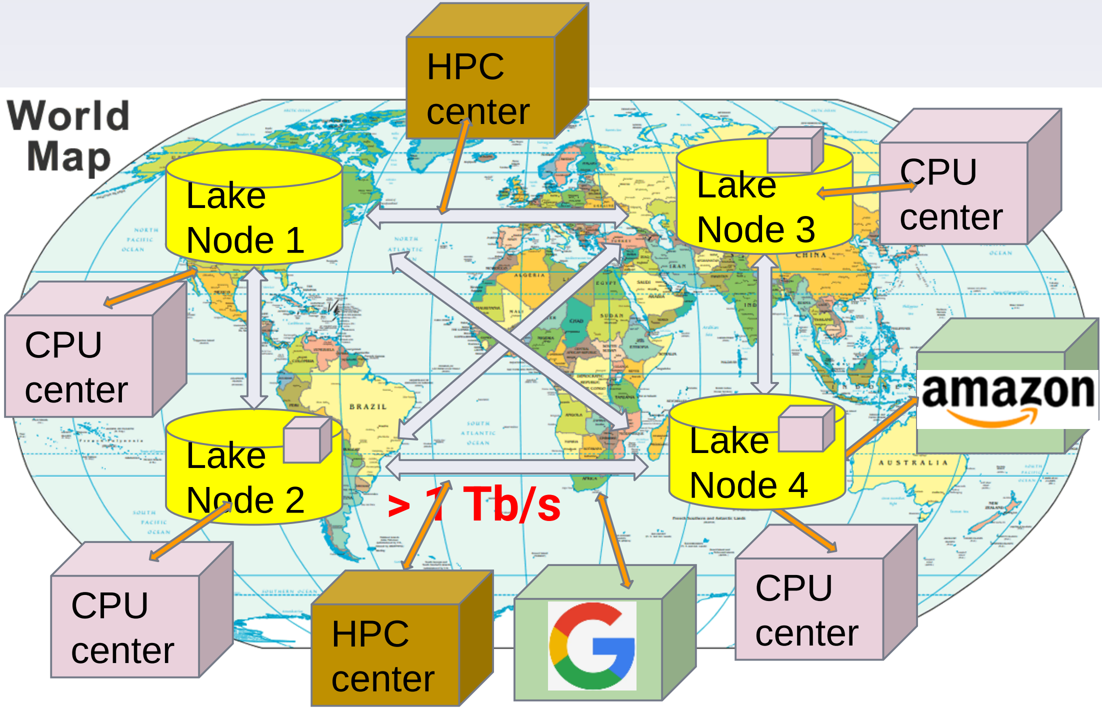
- **YES**, if you care about your data safety (and your capability to access it)
- **NO**, if you can use stateless resources (CPUs!)
  - They come and go fast
  - You can hire them (from a commercial provider, ...)
  - You can use "someone else" resources

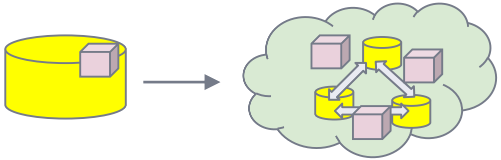


"CPU for free can be found, Disk for free cannot!"

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

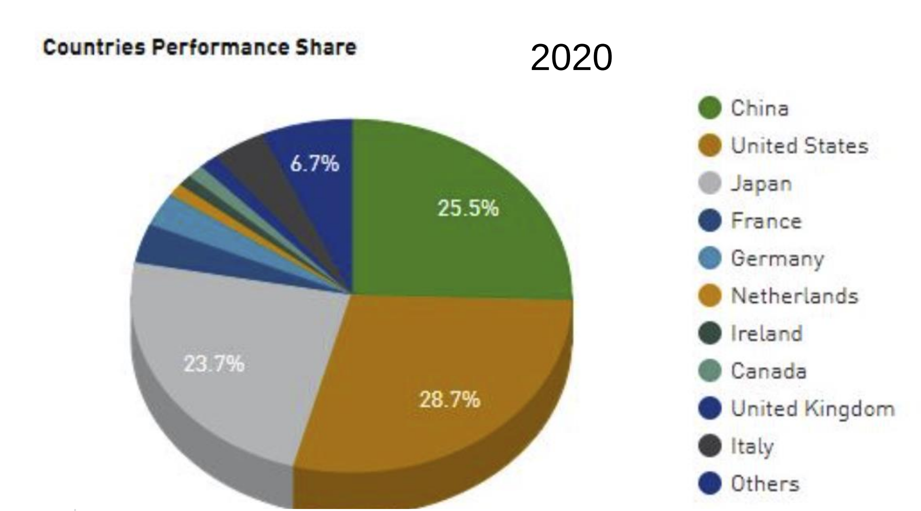
A single genome ~ 100 GB. a 1M survey = 100 PB

CTA projects to 10 PB/y

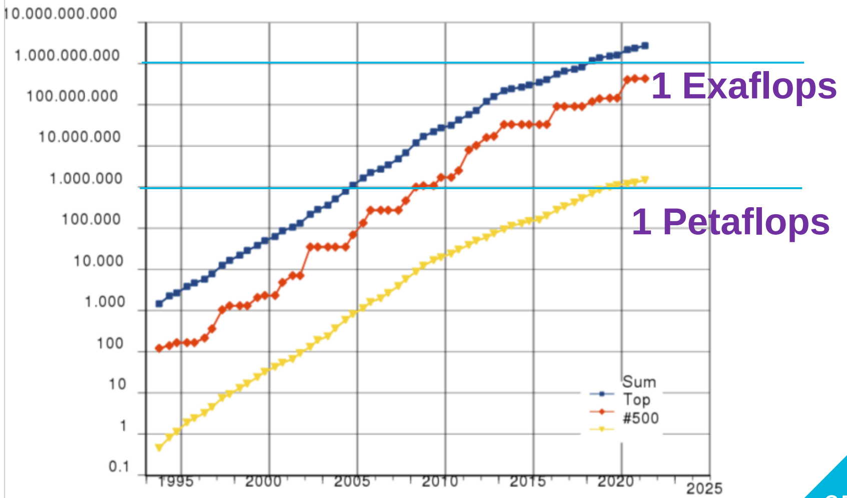
Supercomputing (HPC)

- The world is literally full of Supercomputers. Why ?
    - Real scientific use cases
        - Lattice QCD, Meteo, ...
    - Industrial showcase (“Country XY is technologically capable”)
        - And hence not 100% utilized, opportunities for smart users. Can we be one of them?
- Many not trivial problems to solve:
    - **Data access** (access, bandwidth, ...)
    - **Accelerator** Technology (KNL, GPU, FPGA, TPU, ???, …)
    - **Submission of tasks** (MPI vs Batch systems vs proprietary systems)
    - **Node configuration** (low RAM/Disk, …)
    - **Not-too-open environment** (OS, …)

- Some hint of global slowing down, but not for top systems where the “war” is on
- **1 Petaflops = 10^{15} floating point operations per second**
- **1 Exaflops = 10^{18} floating point operations per second**


Countries Performance Share

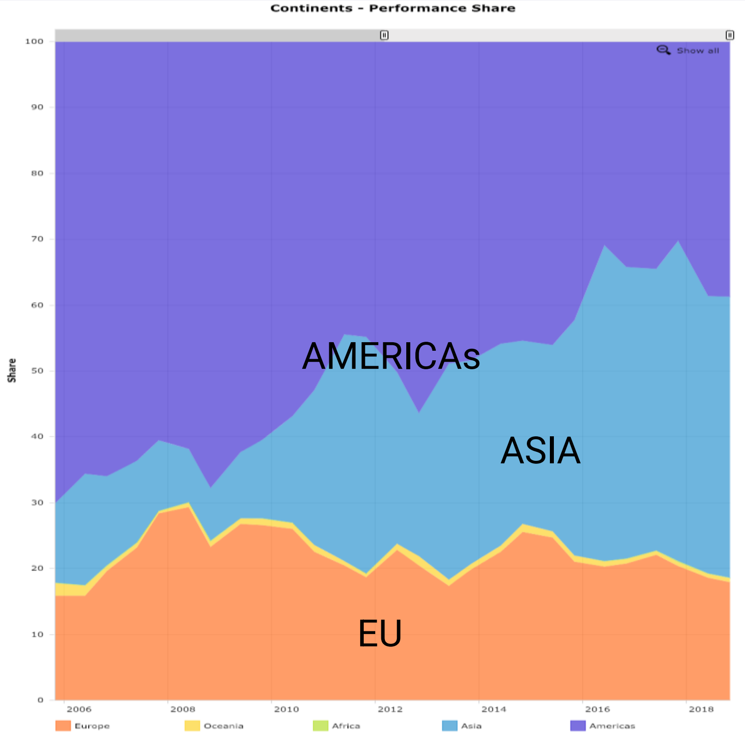
2020

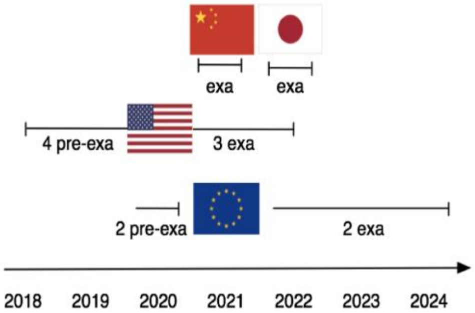

1 Exaflops

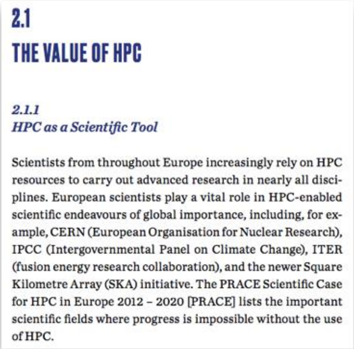
1 Petaflops

Supercomputing - the expected future

- The race will go on, at least between major players
- EU wants to enter the game - never a the top in the last 25y
- Next big thing is **ExaScale** (10^{18} Flops - operations per second)
  - Should be well available by HL-LHC
- Somehow difficult to compare, technologies / benchmarks, but
  - LHC needs today the equivalent of ~30 PFlops
  - A single Exascale system is ok to process 30 “today” LHC
  - **Scaling: a single Exascale system could process the whole HL-LHC with no R&D or model change**
- Some FAs/countries are explicitly requesting HEP to use the HPC infrastructure as ~ only funding; **it is generally ok IF we are allowed to be part in the planning (to make sure they are usable for us)**







2.1 THE VALUE OF HPC

2.1.1 HPC as a Scientific Tool

Scientists from throughout Europe increasingly rely on HPC resources to carry out advanced research in nearly all disciplines. European scientists play a vital role in HPC-enabled scientific endeavours of global importance, including, for example, CERN (European Organisation for Nuclear Research), IPCC (Intergovernmental Panel on Climate Change), ITER (fusion energy research collaboration), and the newer Square Kilometre Array (SKA) initiative. The PRACE Scientific Case for HPC in Europe 2012 – 2020 [PRACE] lists the important scientific fields where progress is impossible without the use of HPC.

US: apparently no way to have a say
EU: ETP4HPC has at least “asked for HEP position”
China/Japan: ???

Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

Pre-Exascale Systems [Aggregate Linpack (Rmax) = 323 PF!] **First U.S. Exascale Systems**

2012 2016 2018 2020 2021-2023

DoE HPC Roadmap: Exascale computing project (2021-2025)

Frontier AMD CPU, AMD GPU; HIP

ORNL

Perlmutter AMD CPU, Nvidia GPU; CUDA

NERSC

Aurora Intel CPU, Intel GPU; SYCL

Argonne

LLNL
IBM BG/Q

EXASCALE COMPUTING PROJECT

LANL/SNL
Cray/Intel Xeon/KNL

LLNL
IBM/NVIDIA

LANL/SNL
TBD

LLNL
TBD

The rest of the world?

MARENOSTRUM 5

Barcelona acogerá el próximo superordenador europeo

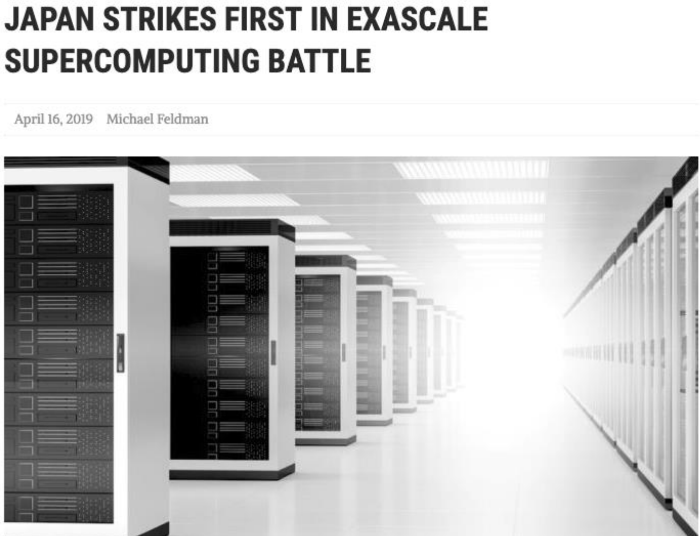
38

• La Comisión Europea financiará con 100 millones de eur construcción y mantenimiento de la máquina. La instalac finales de 2020

Switzerland contributing to one of the most competitive supercomputers in the world to be placed in Finland

Lugano, 2019-06-06 - The Swiss National Supercomputing Centre CSCS of ETH Zurich will represent Switzerland in a joint

L'ITALIA OSPITERÀ UNO DEI SUPERCOMPUTER EUROPEI PRE-EXASCALE



JAPAN STRIKES FIRST IN EXASCALE SUPERCOMPUTING BATTLE

April 16, 2019 Michael Feldman

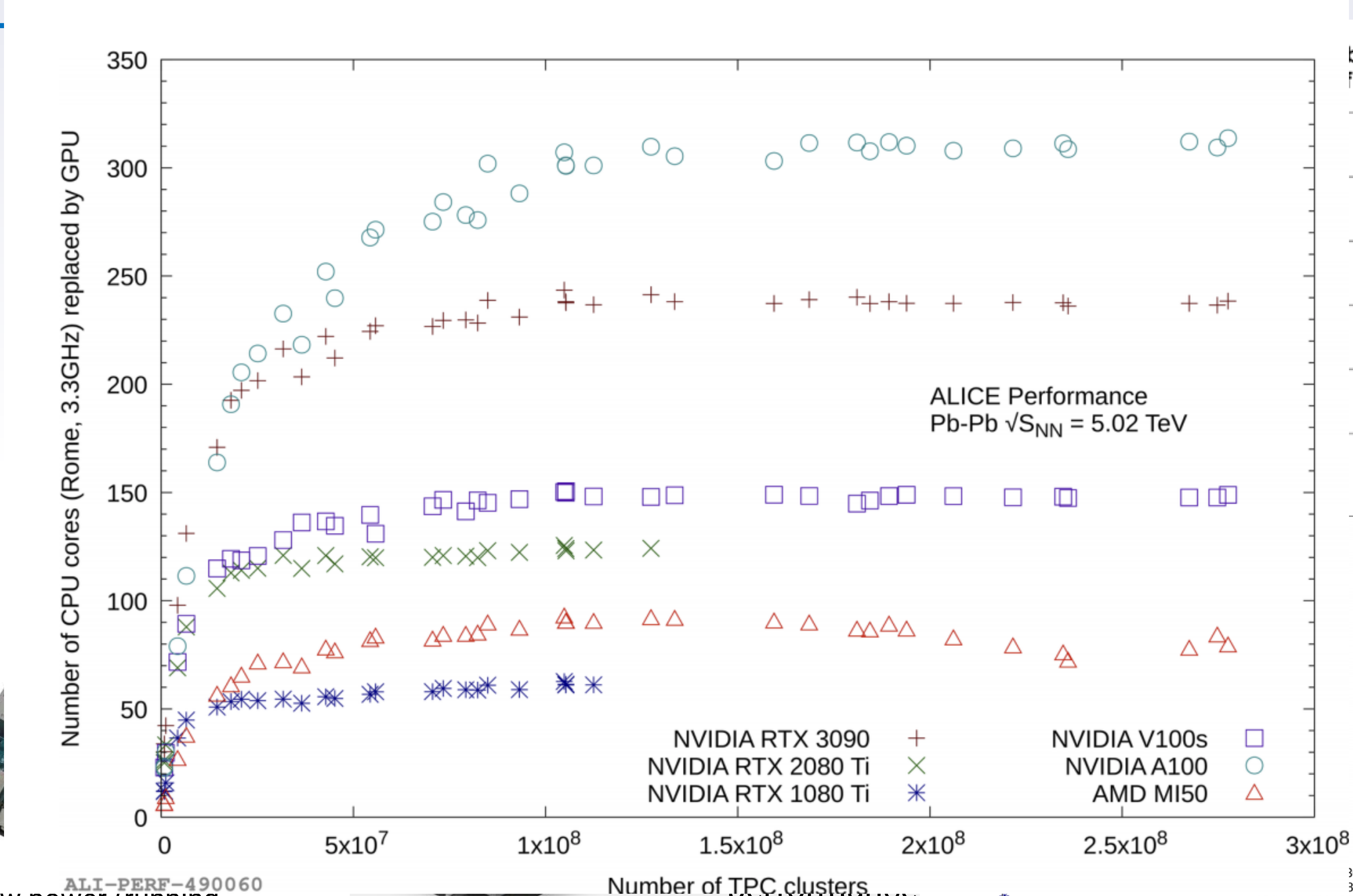
3 EU pre exascale (~250 Pflops) assigned to IT, ES, Finland – Jun 2019 Generally x86 + some GPU

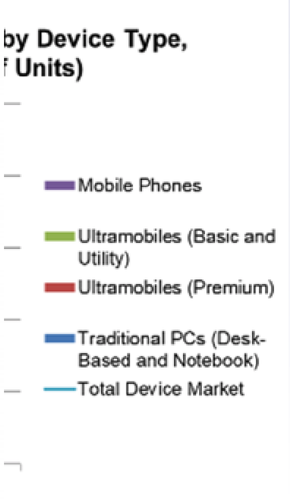
Japan @Exascale by 2022 with Fugaku (ARM (😭) w/o GPU (😀))

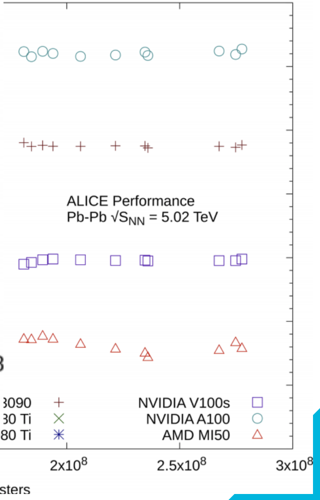
T

-

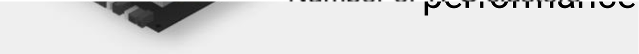
-







Low power (running cost /4)



performance

Physics #1: change analysis model

1. Even today, most HEP physics analysis use a sequential model:
   - Analyze event by event on a single CPU
   - Make it faster by making it embarassly parallel using a lot of CPUs (for example, using the GRID)
   - Big data tools are known to be better at this:
     - **Map&Reduce**: better parallelization, better data distribution
     - **Columnar analyses**: do not work per event, but per category
     - In both cases, you **get away from the event loop**

```
events = load_events()
for ievent in events:
           do_something(ievent)
           do_something_else(ievent)
           accumulate_results(ievent)
Do_final_stuff()
Show_results()
```

What is the expected difference?

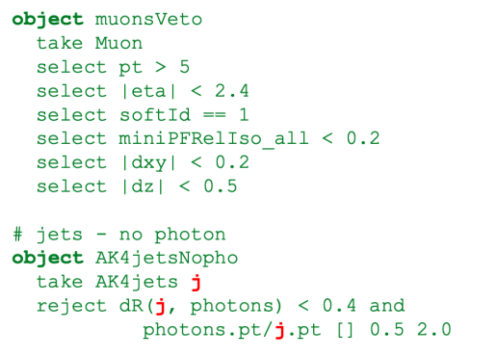
- This is not finding new resources, it is just trying to use better what we have
  - Matches better the underlying hardware, which can be very different – without users needing to know
  - Can change the perceived behaviour of the system
- Grid/Cloud: it is a **container ship**
  - Process many items at the same time, but the shipping time for a given item cannot be made faster
- Reduction facilities: easier to steer more resources to a single use case
  - High priority tasks can overtake a large fraction of

«These 3000 analysis tasks will be done in 5 days»

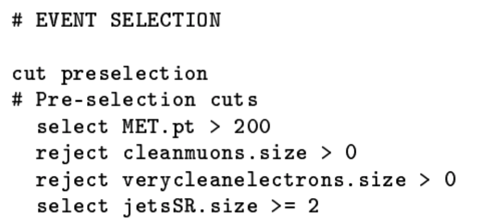
«In the next 5 days you will get an analysis done every 2 min»

Analysis Description Languages

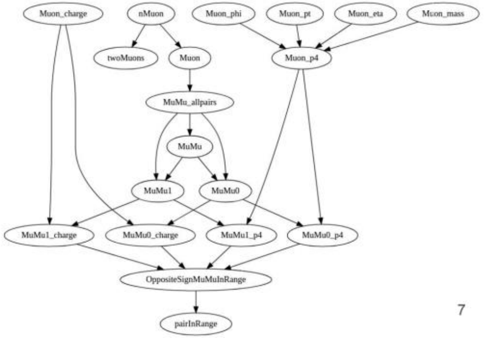
- One "recent" solution which helps abstract from the event loop is the use of Analysis Description Languages (ADLs)
- Describe in "*some high level way*" the analysis, do not write code for that
- Abstract from the actual technology: from the same ADL pseudocode to GRID/GPU/Spark/… optimized code
- Also important for **Analysis Long Term Preservation**: just needs more backends to be maintained



```
object muonsVeto
  take Muon
  select pt > 5
  select |eta| < 2.4
  select softId == 1
  select miniPFRelIso_all < 0.2
  select |dxy| < 0.2
  select |dz| < 0.5

# jets - no photon
object AK4jetsNopho
  take AK4jets j
  reject dR(j, photons) < 0.4 and
           photons.pt/j.pt [] 0.5 2.0
```



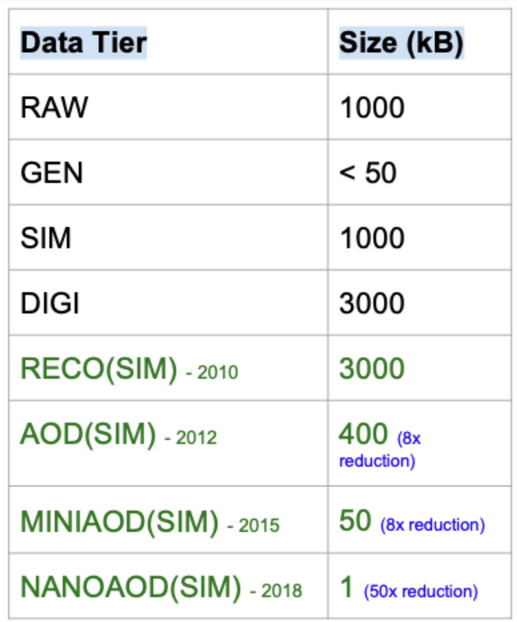
```
# EVENT SELECTION

cut preselection
# Pre-selection cuts
  select MET.pt > 200
  reject cleanmuons.size > 0
  reject verycleanelectrons.size > 0
  select jetsSR.size >= 2
```



Physics #1: change analysis model

2. Re-derivation of physics quantities at analysis time (so in principle repeated N times) is a large source of CPU utilization
    - It is needed because sometimes latest-greatest calibrations are late in the process
    - It is needed if every analysis thinks there is the need to specifically fine tune jet / tracking / id algorithms
    - It costs a lot: CPU to re-run algorithms, Storage to host data samples complete enough to rerun

- As experiment (and their understanding of algos) improve, analysis can be more standardized
    - A single algorithm fits all
    - Calibrations are stable and do not need late second corrections
    - → no need to keep more than 1 algo per object, and to serve large files with low level quantities



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)

"Prevalent analysis format in CMS reduced by a factor 3000x in event size since the start of Run-1"

Use “modern weapons”

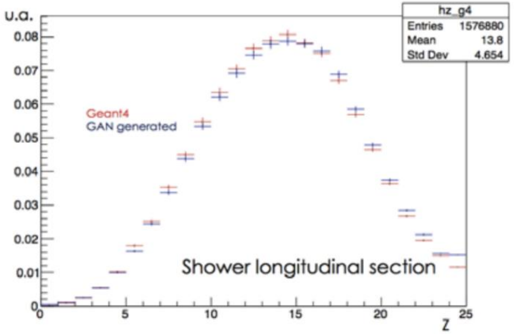
- These can be from the technology point of view (the Big Data Tools we already saw) ..
- … or novel ways to write algorithms. Here obviously AI in general and Machine Learning / Deep Learning techniques stand up
- The space / time here is way too short to go into any detail, but by now ML techniques 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, …)

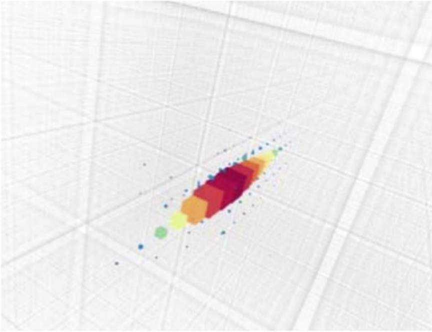
ML usage patterns #1

- 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
- At Simulation level, GANs have shown the potentia to mimic more complex iterative algorithms (like

ng

TensorFlow
LeFlow
FPGA





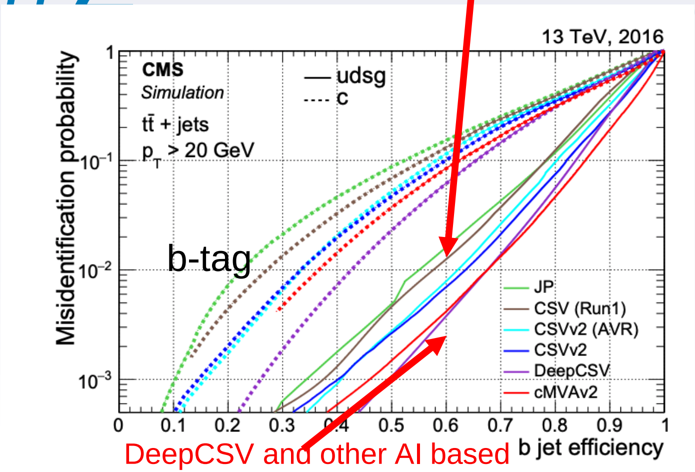
Longitudinal shower shape in a calorimeter from 100 GeV e^- from here. Timing is 1 minute vs 0.04 msec

ML Usage patterns #2

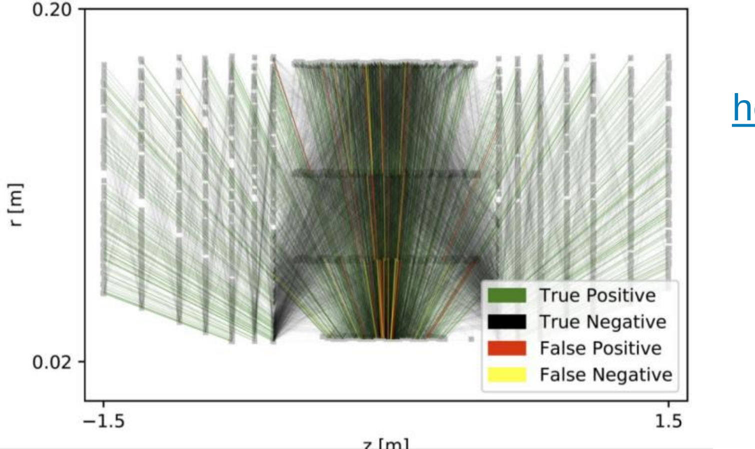
At **reconstruction level**, ML is used generally in two categories of situations:

- **Improvement in classification** (S vs B, and in general category A vs B, C, …) using a large number of (even poorly) discriminating variables
- **Clustering algorithms** which exhibit combinatorial explosion with classical algorithms (jet clustering, tracking)
    - CNNs (input-as-images), Graph Networks

Typical classical algorithm:
60% efficiency for 50x rejection



DeepCSV and other AI based algorithms: 60% efficiency for 300x rejection



here

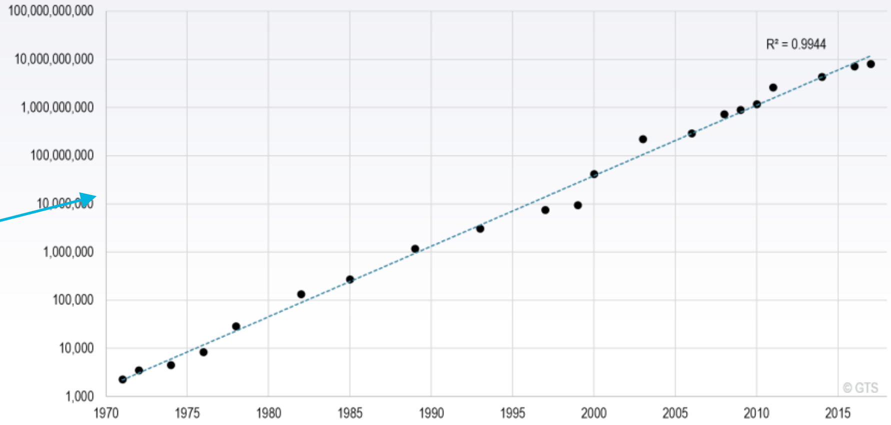
Something **unexpected**…

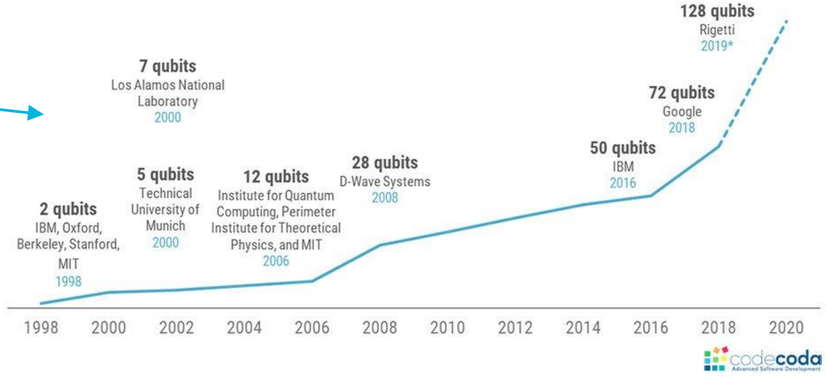
- Well, being unexpected it is difficult to predict, but there are a few options on the table which could be relevant
  - In memory computing
  - FPGAs and CPUs on the same die
  - **Quantum Computing**
  - …
- **A few words just on the last one (if it not too late; then you can look at them offline)**

Quantum computing for HEP?

- QC is promising since in perspective has the potential to overrun classical technologies: we have seen that standard CPUs are improving exponentially (Moore's law)

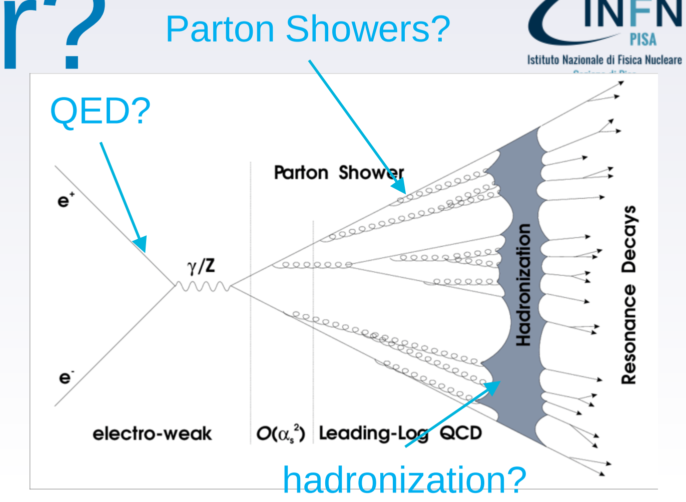
- BUT: QC performance are linked to 2^N, where N is the number of qbits, which is increasing fast → it will beat the CPU exponential sooner than later

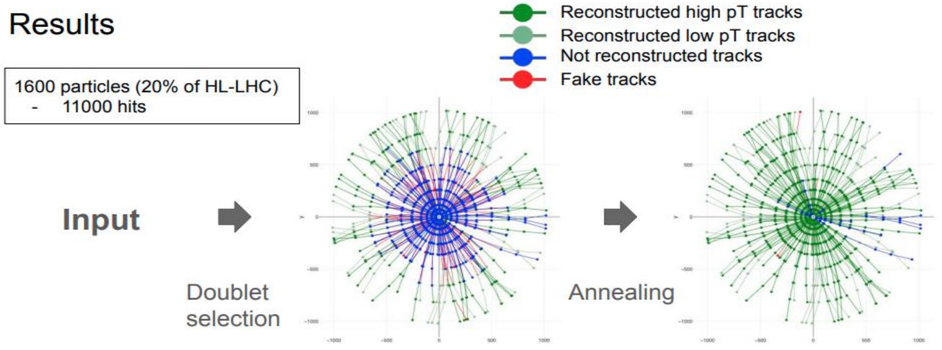




QC in HEP: what for?

- Just a personal list of where it could be useful (next page has links to recent papers)
- **Simulation**: while generating events from pp collisions, we cannot use an exact method, and we are forced to approximations, expansions, … (again LO vs NLO vs NNLO, …). Ideally, a QC system which is made at least locally identical (same H) to a subset of SM could be swapped with
- **Reconstruction algorithms**: use superpos to probe large phase spaces, in particular in algorithms with high combinatorics (e.g. Grc
- **Mimimization**: a generic high dimensional minimization engine, usable in reconstructio analysis, … would be the Holy Grail for us, and could be a drop-in replacement to tools we already know

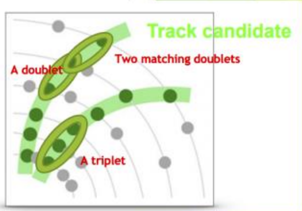



Results

1600 particles (20% of HL-LHC)
- 11000 hits

QUBO: using quantum annealing for pattern recognition

- Given a typical silicon pixel detector and its Hits {H} (in the 10000s), build an "energy" function which is at the minimum when {hits, doublets, triplets} beloging to the same track are considered
  - Then, you can use these as a "seed" for a complete tracking
- Well adaptable to the D-wave formalism, needs a large number of qbits OR a good preselection
- Currently not easily doable: if 10000 hits overall
  - O(10000^2) doublets → QUBO starting from (preselected) doublets
  - O(10000^3) triplets → QUBO starting from (preselected) triplets
  - D-wave: ~1000 not fully connected qubits
- QUBO: quadratic unconstrained binary optimization
  - build an energy function quadratic in the qbits (==doublets, triplets) which has negative term for each pair of {doublets, triplets} which arealigned and from the same track



https://arxiv.org/pdf/1902.08324.pdf

LETTER

doi:10.1038/nature24047

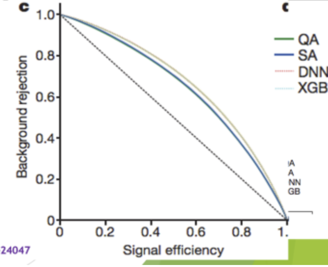
Solving a Higgs optimization problem with quantum annealing for machine learning

Alex Mott[1]†*, Joshua Job[2,3]†*, Jean-Roch Vlimant[1], Daniel Lidar[3,4] & Maria Spiropulu[1]

- First real example of application of QC to HEP (indeed it went to Nature, even if there is no real improvement on any standard Higgs analyses)
- Use quantum annealing (on a D-Wave 1098 qubits) to train a Machine Learning system used in the characterization S vs B in a Higgs search
- Future-proof tested idea: a QC ML training should "one day" be faster. That's it ...
- Use H→ gamma gamma + bkg simulated events to train a ML, 8 kinematic variables + 28 derived quantities
- The quantum system is simulated as an Ising model
- The training output is compared between
  - Quantum Annealing on a D-wave (QA)
  - Simulated Annealing (SA)
  - A Keras Deep Neural Network (DNN)
  - A network built with XGBoost (XGB)
- If you want it only proves the minimization / training works, it does not really prove that it would be any faster with Quantum systems; this is only theoretical at the moment

Table 1 | The kinematic variables used to construct weak classifiers

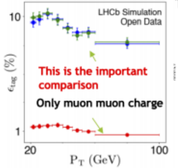
Variable	Description
$p_T^1/m_{\gamma\gamma}$	Transverse momentum (p_T) of the photon with the larger p_T (photon '1'), divided by the invariant mass of the diphoton pair ($m_{\gamma\gamma}$)
$p_T^2/m_{\gamma\gamma}$	Transverse momentum (p_T) of the photon with the smaller p_T (photon '2'), divided by the invariant mass of the diphoton pair ($m_{\gamma\gamma}$)
$(p_T^1+p_T^2)/m_{\gamma\gamma}$	Sum of the transverse momenta of the two photons, divided by their invariant mass
$(p_T^1-p_T^2)/m_{\gamma\gamma}$	Difference of the transverse momenta of the two photons, divided by their invariant mass
$p_T^{\gamma\gamma}/m_{\gamma\gamma}$	Transverse momentum of the diphoton system, divided by its invariant mass
$\Delta\eta$	Difference between the pseudorapidity $\eta=-\log[\tan(\theta/2)]$ of the two photons, where θ is the angle with the beam axis
ΔR	Sum in quadrature of the separation in pseudorapidity η and azimuthal angle ϕ of the two photons ($\sqrt{\Delta\eta^2+\Delta\phi^2}$)
$\|\eta^{\gamma\gamma}\|$	Pseudorapidity of the diphoton system



https://www.nature.com/articles/nature24047

Tensor Networks used as a (quantum) replacement of NNs

- Idea: event categorization (S vs B, "A" vs "B) is a typical problem for us, at many levels
  - Heavy vs light quarks in jets; gluons vs quarks in jets; true vs fake tracks; leptons vs pions, ...
  - Here: b or \bar{b} as the initiator of an hadronic jet @ LHCb
- One typical solution is via Dense NNs, with as many inputs as possible, even those with mild discriminating power
- Alternative Quantum-inspired (possible to be deployed on QC? *Maybe*...) are Tensor Networks; a recent paper shows that at the very least is not worse th izing events from b and \bar b events @ LHCb

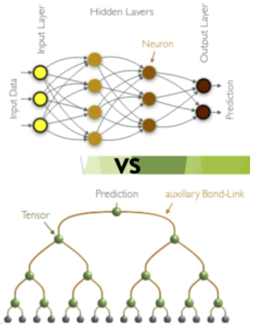


This is the important comparison

Only muon muon charge

Added values:

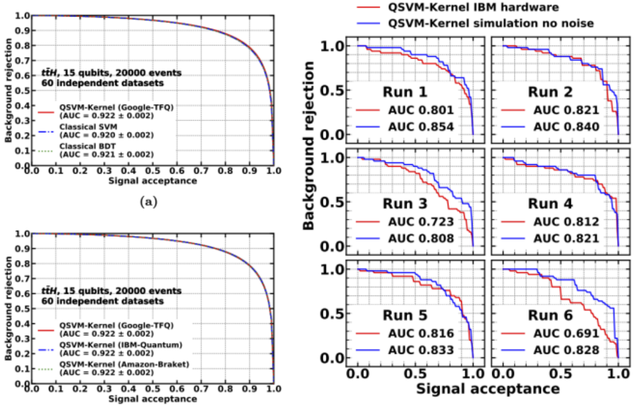
- easy to compute what in ML we would call confusion matrices
- Easy to prune the structure in a tuning speed / accuracy
- (but same is true for classical ML...)



47

https://arxiv.org/pdf/2004.13747.pdf

Application of Quantum Machine Learning using the Quantum Kernel Algorithm on High Energy Physics Analysis at the LHC



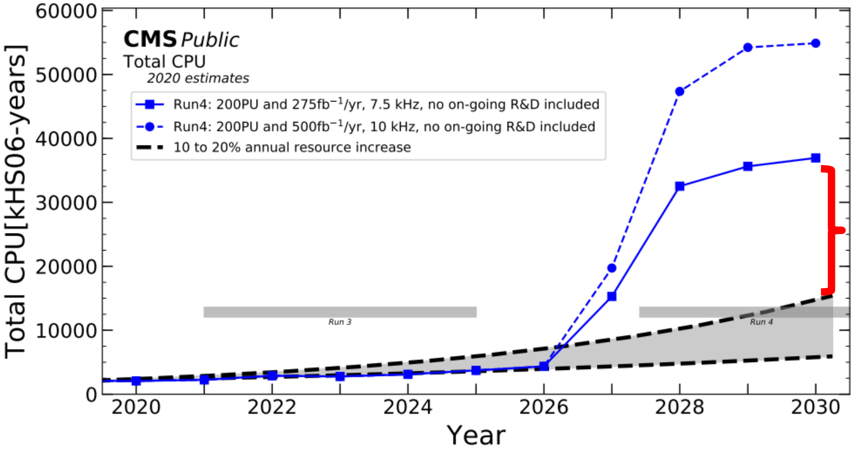
(a)

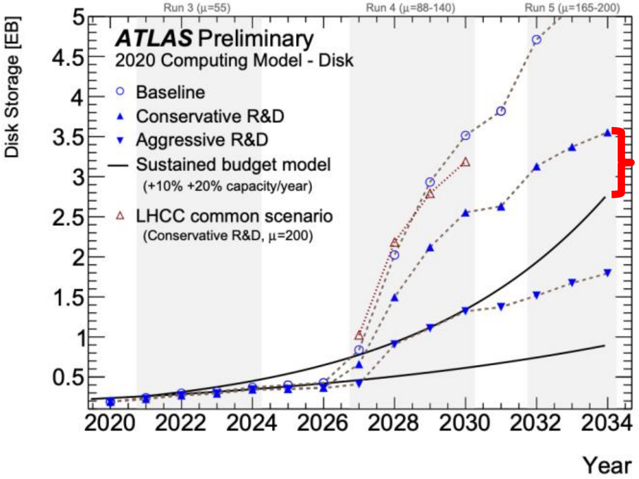
(b)

https://arxiv.org/pdf/2104.05059.pdf

Current status from experiments

- In the last ~5 years the modelling for the needs for HL-LHC have evolved, in large part following some of the “recipes” we listed
  - Infrastructure changes: datalake, fewer copies of data on storage
  - Physics changes: smaller data formats, less CPU for analysis





Missing factors ~2-3x (and in some cases we are already there)

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 (big data, quantum, data lake)