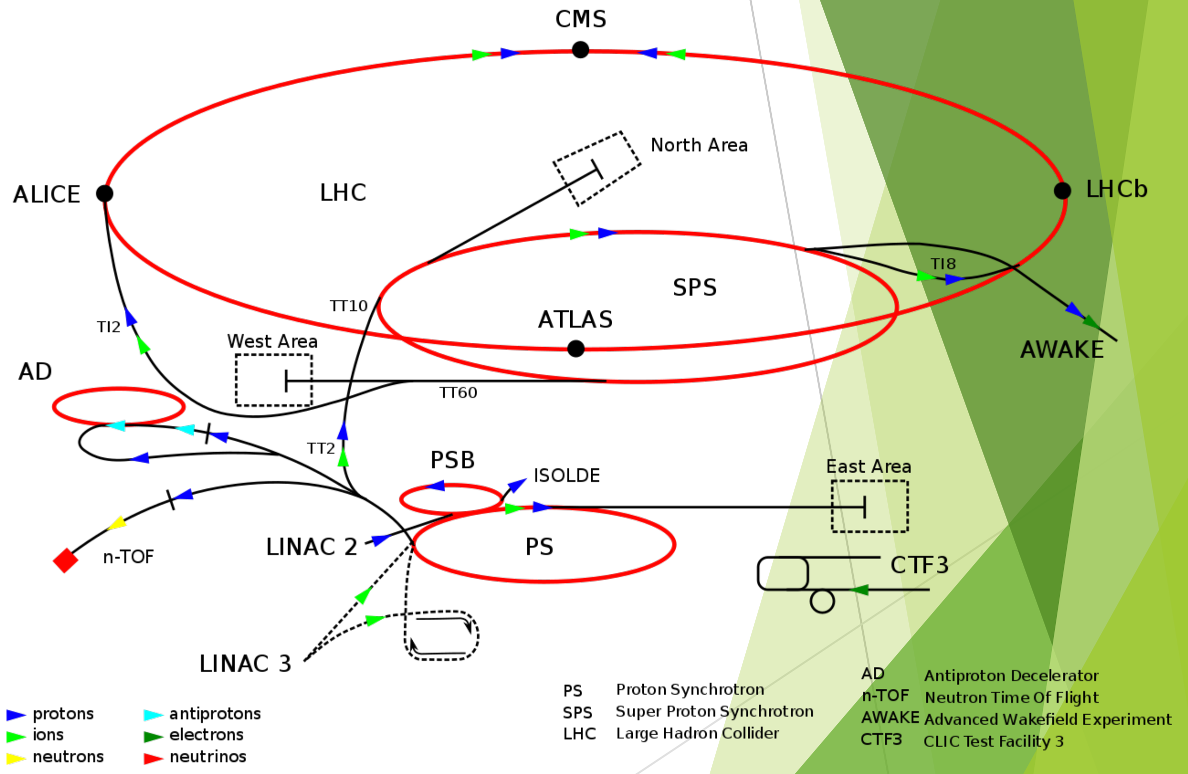
High Luminosity LHC - Computing Models and Impact of Quantum Computing

Tommaso Boccali (INFN Pisa)

What is High Luminosity LHC (HL-LHC?) #1

- Let’s start with some LHC numbers (for computing)
- LHC = Large Hadron Collider
  - Operating today @ 13 TeV, top 2 10^{34} cm^{-2} s^{-1} instantaneous luminosity via pp collisions bunched @ 25 ns
  - Designed for a vast physics program; clearly the discovery / exclusion of the **Higgs boson was top in the list**
  - This means, given a total inelastic cross section of ~100 mb, **35 collisions per bunch crossing** averaged along O(10) hour fills
  - If we naively consider that the big detectors have ~100M acquisition channels (assume 1 byte/channel), the VIRGIN data rate of the big detectors (ATLAS, CMS) would be **4 PB/s**



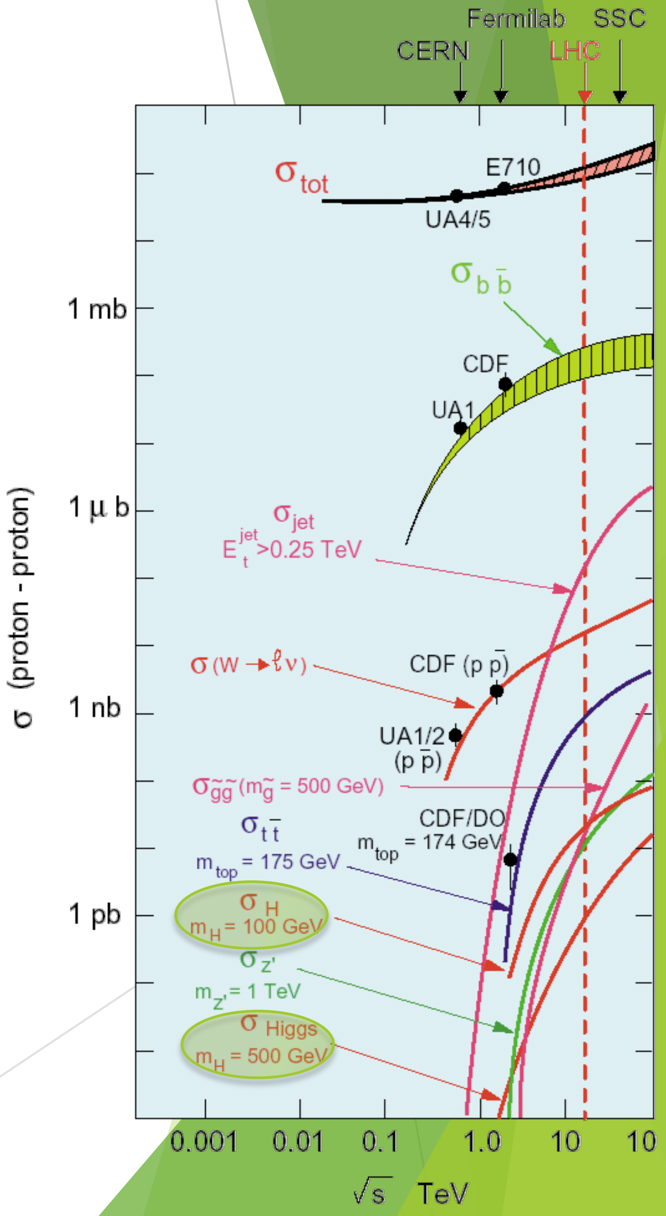
Higgs boson production, expected mechanisms at LHC planning times

- Higgs production cross section (how probable to create one) increases very sharply with collider energy
- The actual number of produced events in a given process is proportional to its cross section, and the collider luminosity
- **$N = \sigma \times L_{int}$**

How probable the process is “per collision” (1 m^2 = 10^{28} barn)

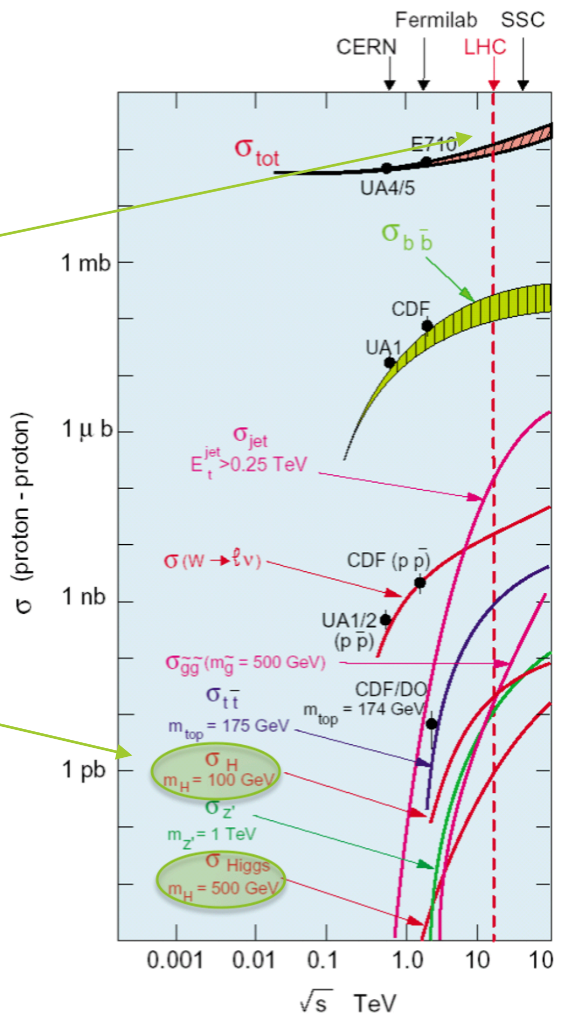
How many collisions we are trying m^{-2}

- where L_{int} is the integrated luminosity an experiment has been given
- The Higgs production cross section is ~50 pb @ a 13 TeV collider
  - @ 1 TeV collider it would be ~ 100-1000 times lower, this is the reason why a direct positive discovery at TeVatron was not probable



Putting all together …

- If your goal is to have some million generate Higgs boson in a ~ 5 y run period, you need to integrate (per exp) some 100 fb^{-1}
- 5y are (accounting for LHC availability, shutdowns, etc) ~ 30 Msec collision time
- So, you need **O(10^{34}) cm^{-2} s^{-1} instantaneous Luminosity**
- The problem: you cannot produce this without producing this
- **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)**
- **But: the very same parameters drive to the data flux O(PB/s) → we have a computing problem!**



How to solve it?

- There are easy handles to implement
  - Do not read all the detectors, read only “channels above noise” → 100x reduction
- Select and save only interesting events, drop the rest (“the triggers”)
  - Dangerous, you can bias the sample
  - Difficult, the higher the number of superimposed events, the smaller are differences
- This is history by now, with LHC in operations, and Higgs discovery has been possible with outgoing rates to “offline” of
  - 1-2 kHz of events
  - 1-2 GB/s of data
- These data need to be stored, processed, analyzed and compared with a similar amount of Monte Carlo Simulations

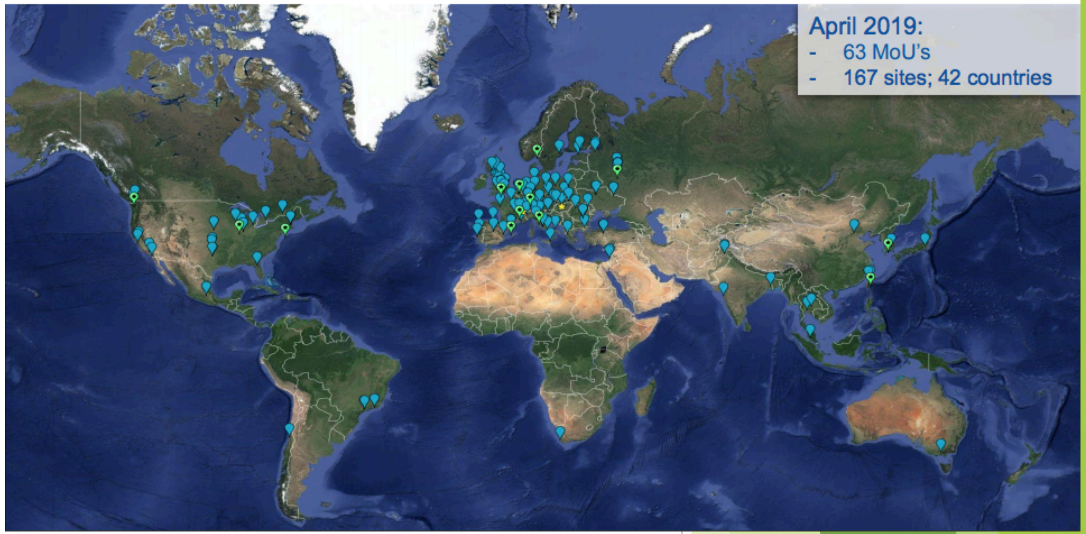
Today's LHC Computing is …

- The largest scientific computing system
- The largest DISTRIBUTED computing system
  - By farm the largest GRID deployment, ~ 200 sites
- The highest scientific network utilizer:
  - > 100 GB/s moved at any moment
- The largest repository of scientific data (over 1 Exabyte overall)

- … and it works, so why bother?

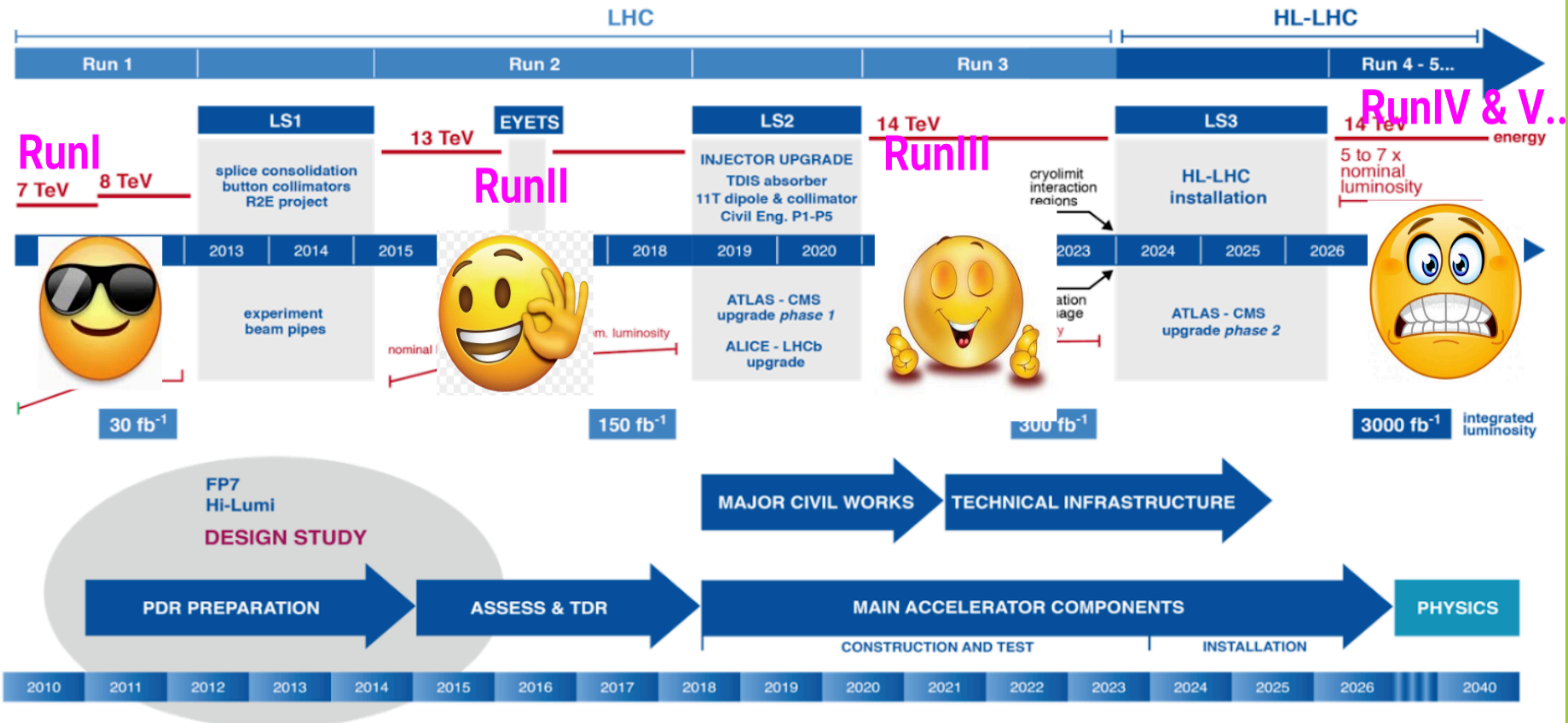
Worldwide LHC Computing GRID

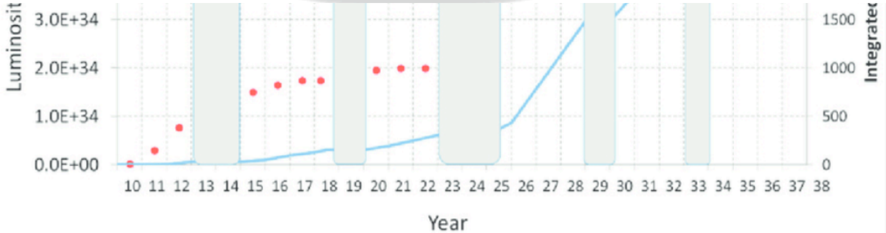
WLCG Collaboration



April 2019:
- 63 MoU's
- 167 sites; 42 countries

Experiment	CPU (kcores)	Disk (PB)	Tape (PB)
ALICE	100	100	85
ATLAS	280	230	310
CMS	200	160	280
LHCB	45	45	90
TOTAL	**625**	**535**	**765**





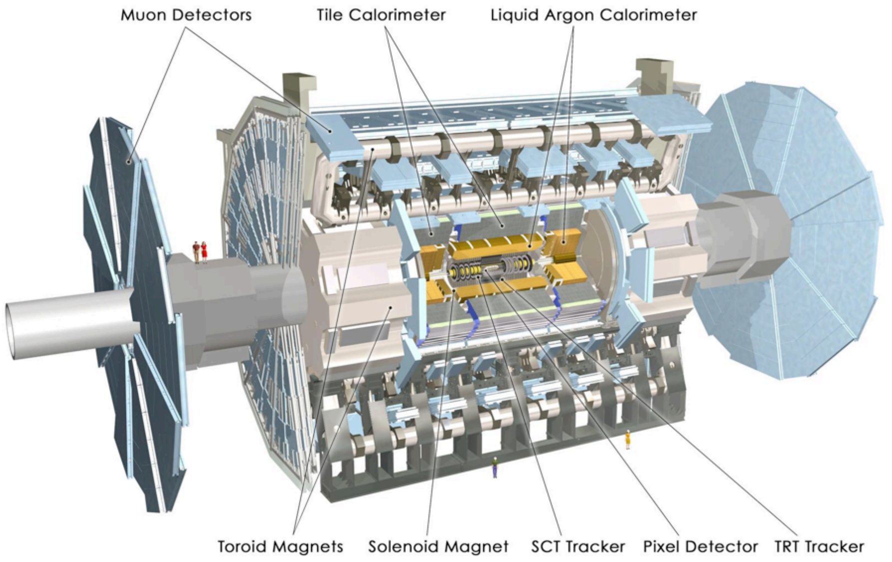
(to be clear: I am not even considering RunIII, it is just a "simple" extension of RunII for ATLAS and CMS - no tension)

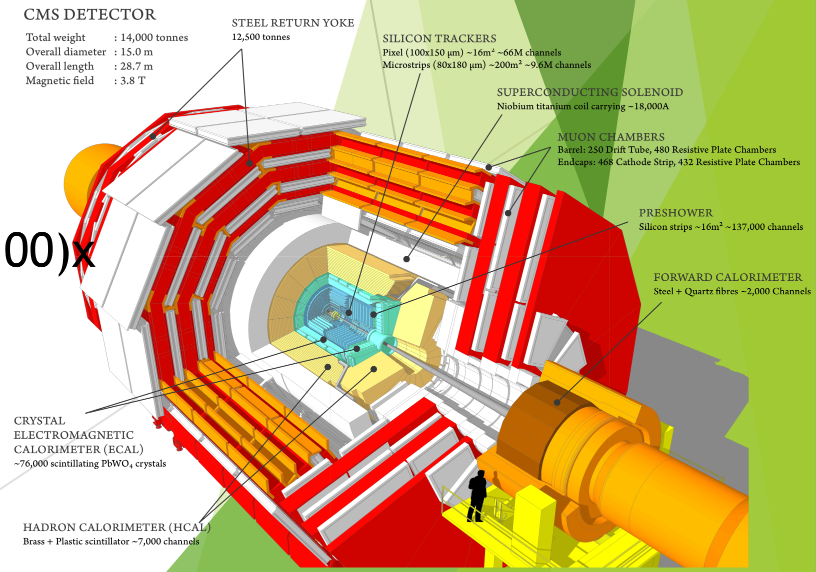
CMS and ATLAS computing scaling @ HL-LHC

- **# events collected/y** = Experiment live time * Experiment rate to offline
  - LHC RunII: 7 Ms/y * 1000 Hz = ~ 7 B events/y
  - LHC RunIV: 7 Ms/y * 7.5 kHz = ~ 50 B events/y
- **Bandwidth, total storage** = # events collected * (1+ f_{MC}) * typical_event_size
  - f_{MC} ~ 1-2
  - Typical event size:
    - LHC RunII: 1 MB/ev
    - LHC RunIV: 5-10 MB/ev
- **Computing power** = # events collected * (1 + α*f_{MC}) * F(event_complexity)
  - F(event_complexity) usually superlinear in instantaneous luminosity
  - α: how much more expensive is to process a simulated events with respect to a real data one. O(2) < α < O(20+)
- Storage is also ~ integral with time
- $\text{Storage}_{\text{YearN+1}}$ = $\text{Storage}_{\text{YearN}}$ + $\text{Delta}_{\text{NEW EVENTS}}$

~ 7.5 * 10 → O(50-100)x
for storage

~ 7.5 * 10 → O(50-100)x
minimum
for CPU



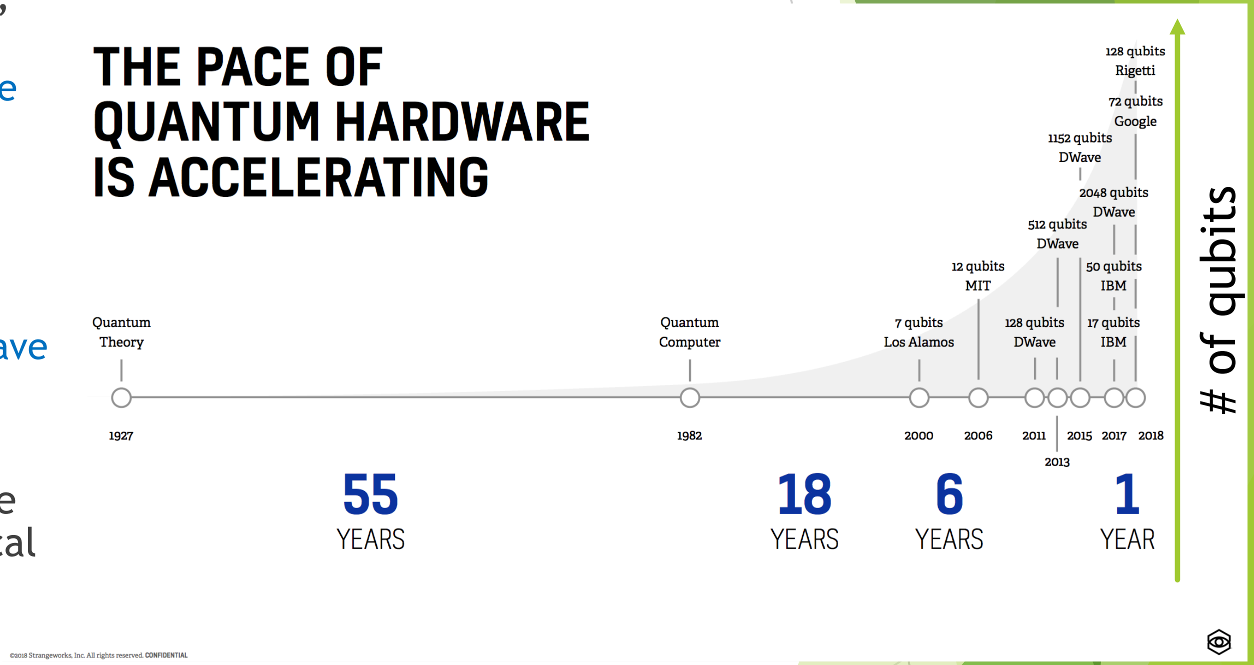


So, it works today but…

- A simple extrapolation @ HL-LHC (without any model change) easily gives factors 50-100x (“Billions Eur per year”)
- Clearly, a lot can be done, with the masterplan as of now on
  - Reducing overheads and inefficiencies (fewer copies of data, fewer simulations required, fewer processing passes, …)
  - Acquiring new “cheaper” resources (Supercomputer Centers, Commercial Clouds, …)
  - Using “cheaper” systems (GPUs, Tensor Processors, FPGAs, …)
  - Using “faster algorithms” (Artificial Intelligence in its various declinations, …)
  - …
- Baseline: we are currently not at the level of being able to guarantee HL-LHC at the same price as today, **but we are getting closer and closer…**

Where is the "space" for quantum computing here?

- There are parts of our workflows which would "naturally fit"
  - parton parton interactions → it is a quantum system, we use classical simulations just because we do not have anything else
- Solving the problem (== fitting a budget) is obviously not optimal for us
  - You can always do things better / in an easier way if you have more
- We are somehow worried by the trends (the double exponential in # of qubits and capabilities per qubit), and we do not want to be found unprepared in case of a technological breakthrough



THE PACE OF QUANTUM HARDWARE IS ACCELERATING

- Let's see where / how Quantum Computing would fit in our processing model

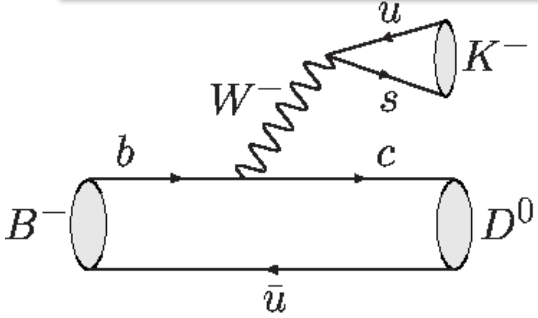
Reality

«nature»

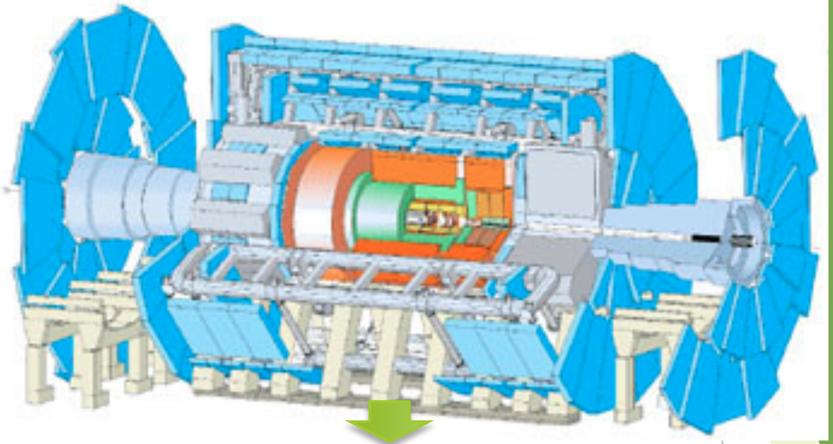
LHC collisions

«nature»

Decay of unstable particles



ATLAS

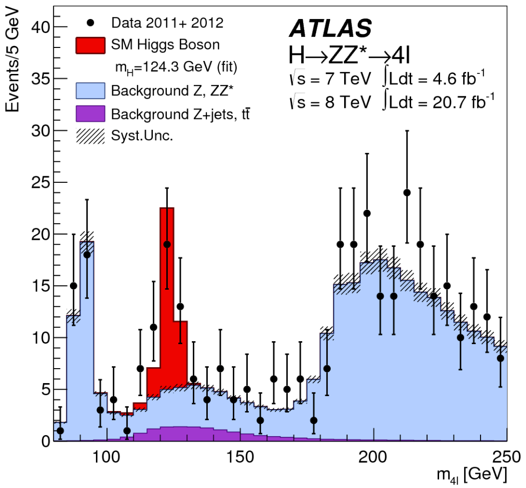


Detector electronics — HW/SW

Trigger (selection) — HW/SW

Reconstruction — SW

Analysis — SW



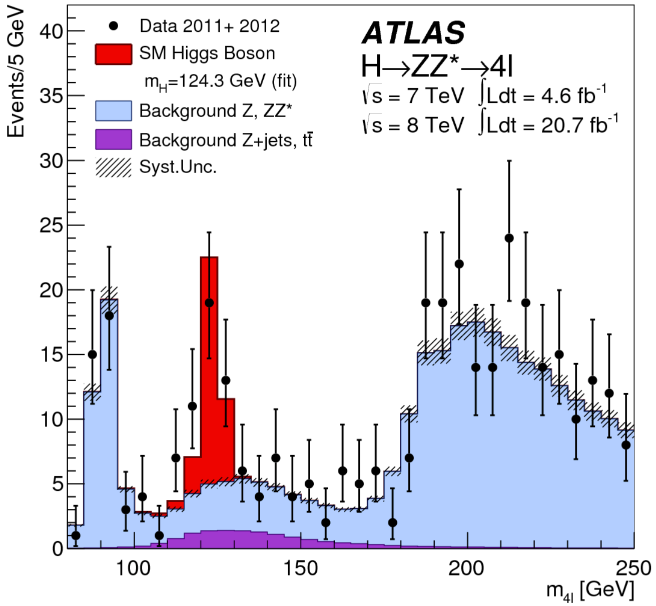
Simulation - all SW

- Theoretical model (“generators”)
- Simulation of decays of unstable particles
- Simulation of interactions particle-detector
- Simulation of detector electronics
- Trigger Simulation
- Reconstruction
- Analysis

GEANT4
A SIMULATION TOOLKIT

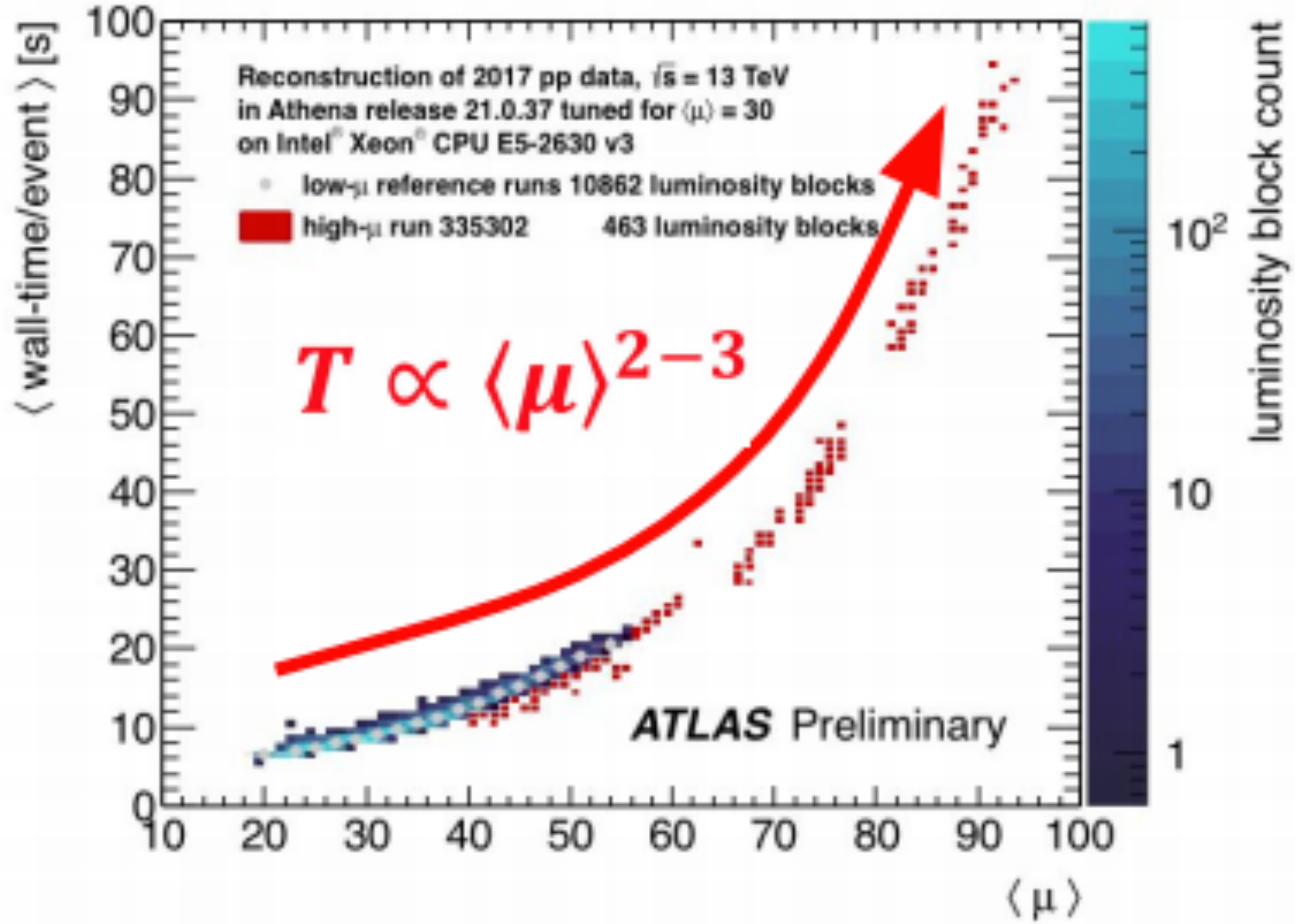
$$L_{QCD} = \sum_q \overline{\psi}_q \left(i\gamma_\mu D^\mu - m_q \right) \psi_q - \frac{1}{2} Tr\left[\overline{G}_{\mu\nu} \overline{G}^{\mu\nu} \right]$$

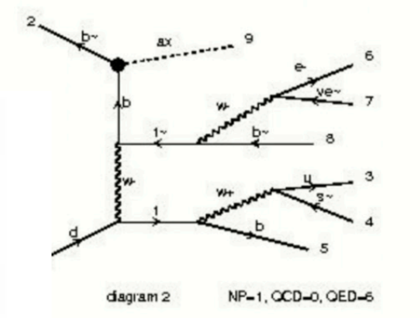
PYTHIA

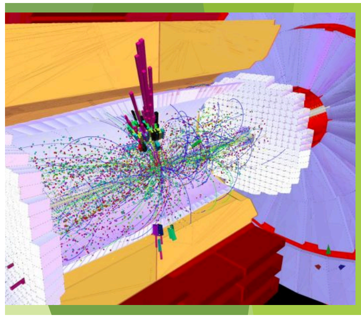


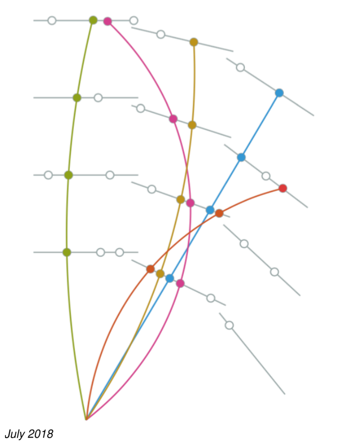
What are the typical algorithms doing?

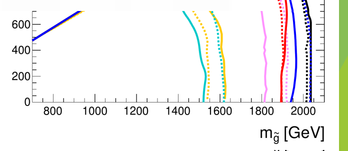
- **Generation** is th
  modelling of a c
  sequential steps
  - Currently, do
    loops and legs
- **Simulation** in **G**
  interactions par
  - Some of them
  - The more the
- **Reconstruction**
  combinatorial a
  - Searching for
- **Analysis** is … an
  - In general, th











Is QC another “weapon” we should study?

- **Disclaimer: we are here mostly in the initial learning phase; our understanding of QC possibilities is not necessarily adequate**
  - **A very honest answer would be “we do not know yet”**
- Bird’s eye evaluation:
  - **Quantum simulation** could in principle take the place of algorithmic generators, at least for some specific processes
  - **Quantum computing** could be used in principle for generic minimizations, or in order to speed up combinatorial algorithms
    - Or in principle ANY algorithm via a Grover approach
- Let’s start from the problems we see, and then direction….

Problem 1 - the data problem

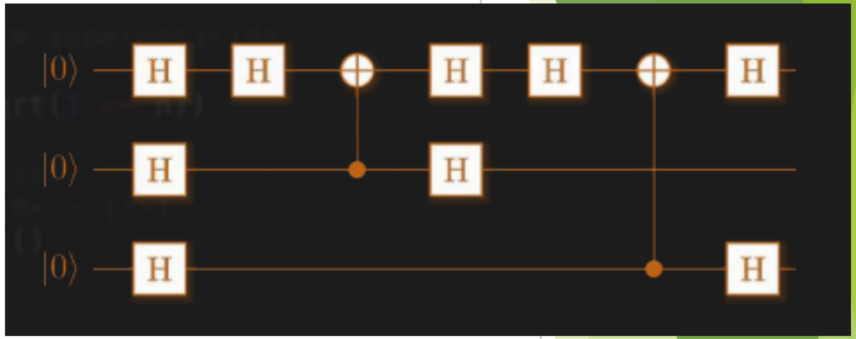
- From what we described up to now, it is clear that we need to access / crunch / move large data amount during our processing; while typical QC examples we have seen are the factorization of a prime number (~ **0 bandwidth**)

- **Example**: expect a 10 MB event to be processed in Reconstruction (in 2027), in some 50 seconds on a CPU→
  - Currently QC entangled states are usec-msec long – so necessarily computation would be this fast
  - But which is the time needed to “prepare the state”? How fast must the data be moved to the system?
    - Who do you “move” 10 MB of information on the quantum system?
- **Which is the bandwidth we can expect from a quantum computer?**
  - Even if processing is fast (say quantum tracking), what if it takes 10 min to create the initial state?
    - We never really got an answer by the technology guys …
- **Can QC be imagined applicable to algorithms which operate on real data at all?**
  - For these, the above bandwidth requirements raise by orders of magnitude (a 10 MB event to be processed EVERY few msec) → **it would NOT be the first place where I try and use QC…**

Problem 2 – the programming model

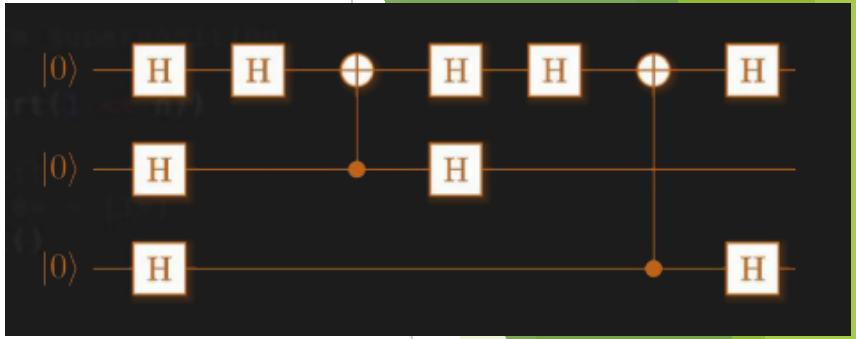
- Our software development model uses mostly C++, with some CUDA sneaking in recently
- This is completely different from gate programming on a QC
- I recently had a Google/Cirq course @ CERN: one day to program an Hadamard gate
    - Difficult to see how to go from there to "particle tracking"
    - "it is not our job"

- **Luckily things are getting better …**



Software tools / libraries

- Quantum circuits are described in terms of gates and transitions, and depend a lot on the internals of the QC
- But we also have:



IBM QisKit

ProjectQ

Powerful open source software for quantum computing

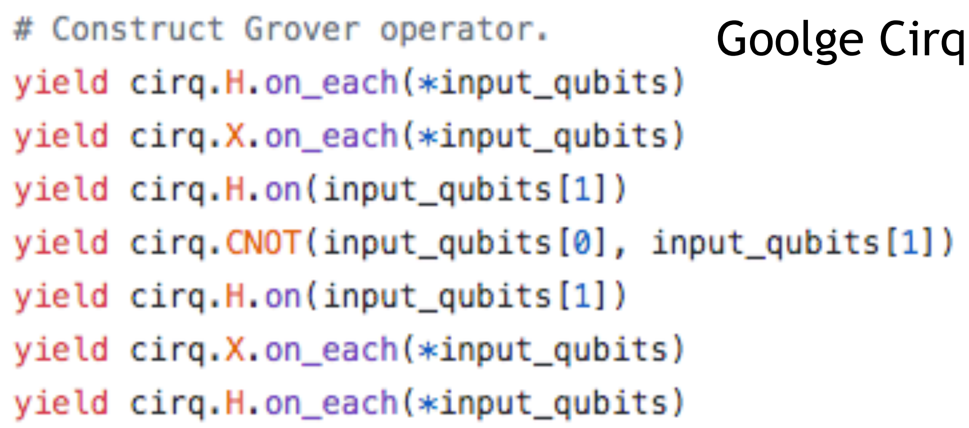
Cirq

The Q# Programming Language

Microsoft

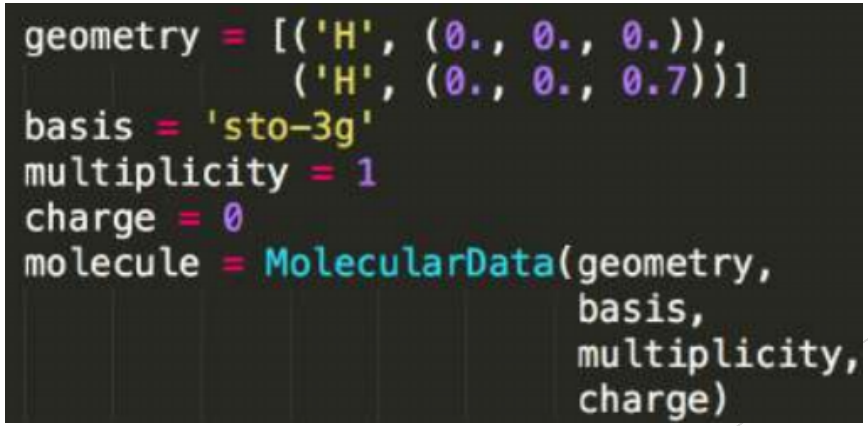
Either low or high level

- You can either really program qubits, operations, measurements, for example if you want to program a Grover Oracle directly
- But in real HEP use cases, you want to use much higher level constructs
  - Possibly the same code / algorithms you use today
  - «put a H atom in (0,0,0) and another in (0,0,0.7) and let the system (H_2) evolve» → get energy on lower state, for example (chemistry)
  - Nothing available today for physics …
    - → clear target for joined development



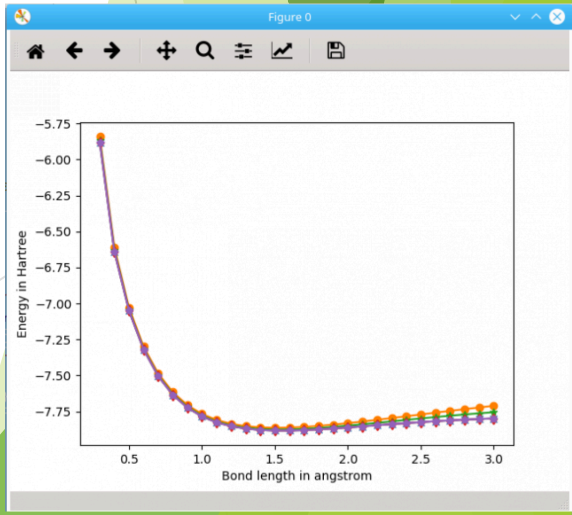
```
# Construct Grover operator.
yield cirq.H.on_each(*input_qubits)
yield cirq.X.on_each(*input_qubits)
yield cirq.H.on(input_qubits[1])
yield cirq.CNOT(input_qubits[0], input_qubits[1])
yield cirq.H.on(input_qubits[1])
yield cirq.X.on_each(*input_qubits)
yield cirq.H.on_each(*input_qubits)
```

Goolge Cirq



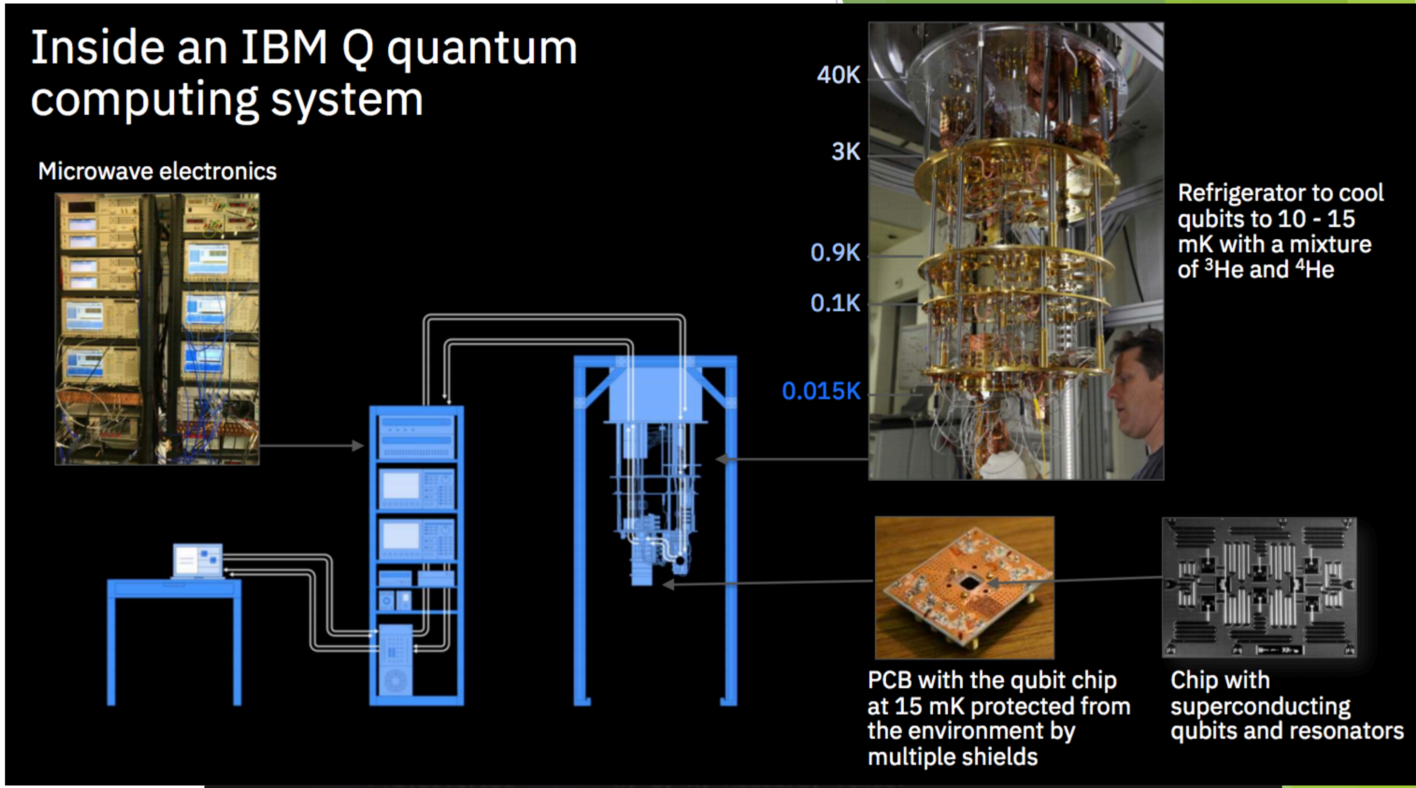
```
geometry = [('H', (0., 0., 0.)),
            ('H', (0., 0., 0.7))]
basis = 'sto-3g'
multiplicity = 1
charge = 0
molecule = MolecularData(geometry,
                         basis,
                         multiplicity,
                         charge)
```

ProjectQ



Access to resources

- Difficult to think that a standard WLCG computing center will host a QC «soonish»
  - mK setup, em shielded
- If you buy one, it will obsolete by the time it gets delivered
- Much more reasonable to imagine a continued / shared / pay-per-use Cloud level access to remote resources
- Already available today (Google, IBM, Rigetti, …). Has also the advantage to shield the users from actual remote setup
  - Emulator or a real QC hardware?



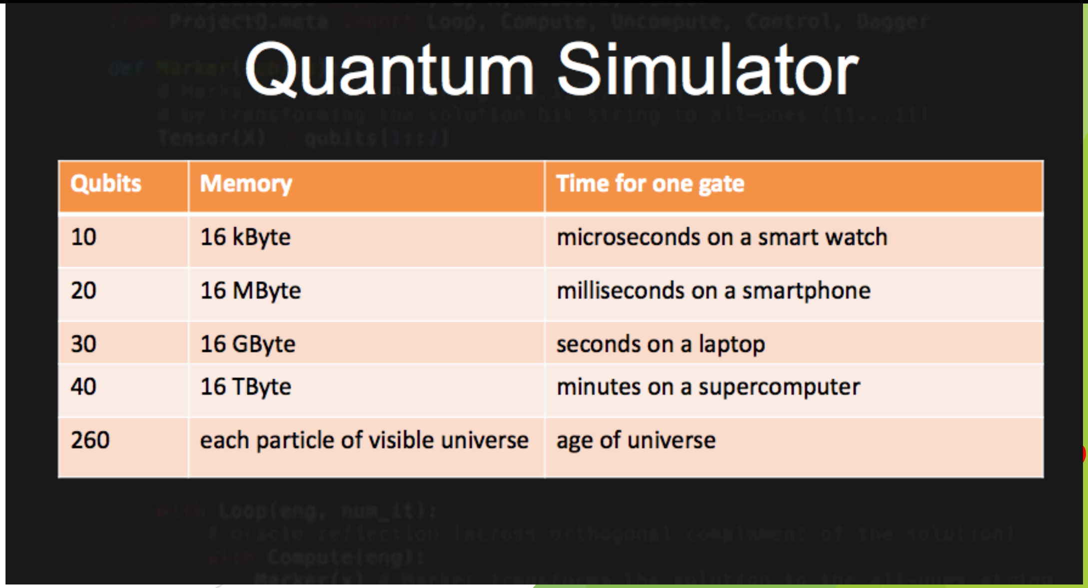
Inside an IBM Q quantum computing system

Microwave electronics

Refrigerator to cool qubits to 10 - 15 mK with a mixture of ^{3}He and ^{4}He

PCB with the qubit chip at 15 mK protected from the environment by multiple shields

Chip with superconducting qubits and resonators

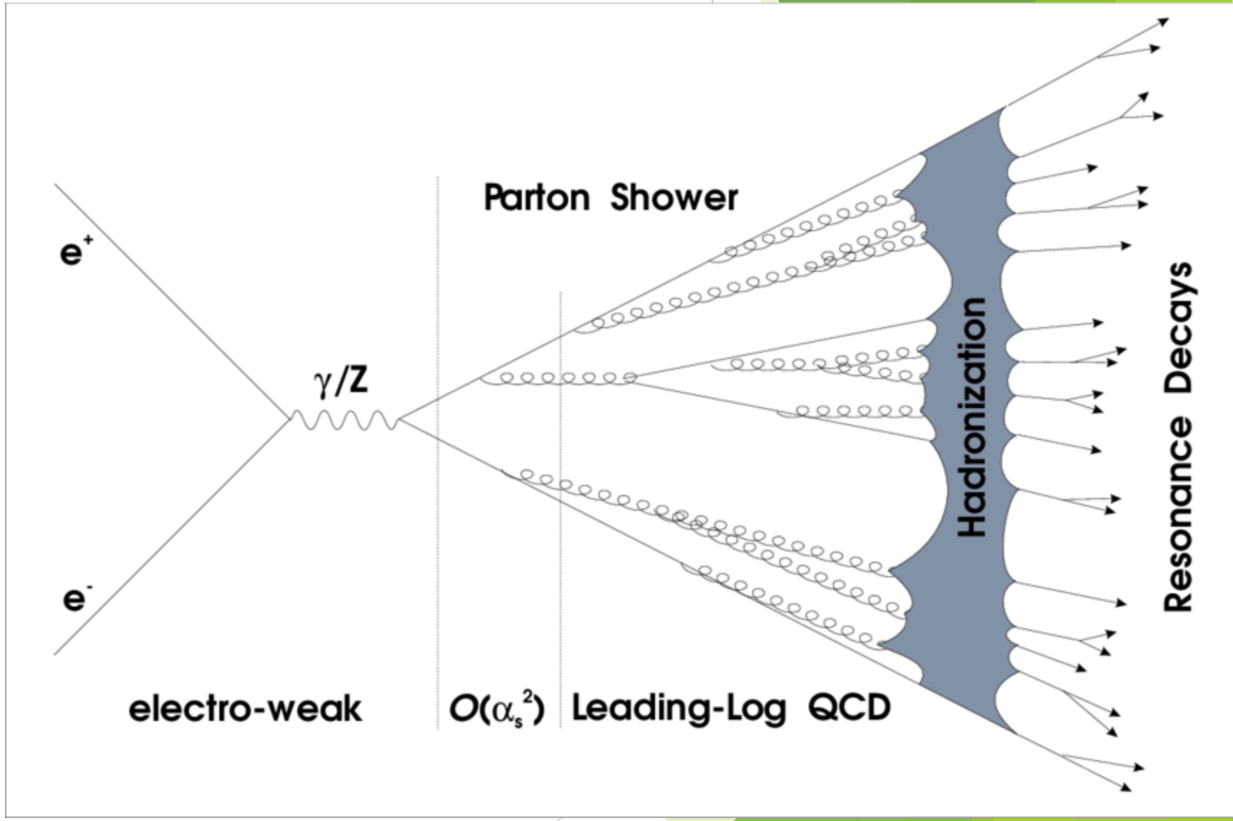


Quantum Simulator

Qubits	Memory	Time for one gate
10	16 kByte	microseconds on a smart watch
20	16 MByte	milliseconds on a smartphone
30	16 GByte	seconds on a laptop
40	16 TByte	minutes on a supercomputer
260	each particle of visible universe	age of universe

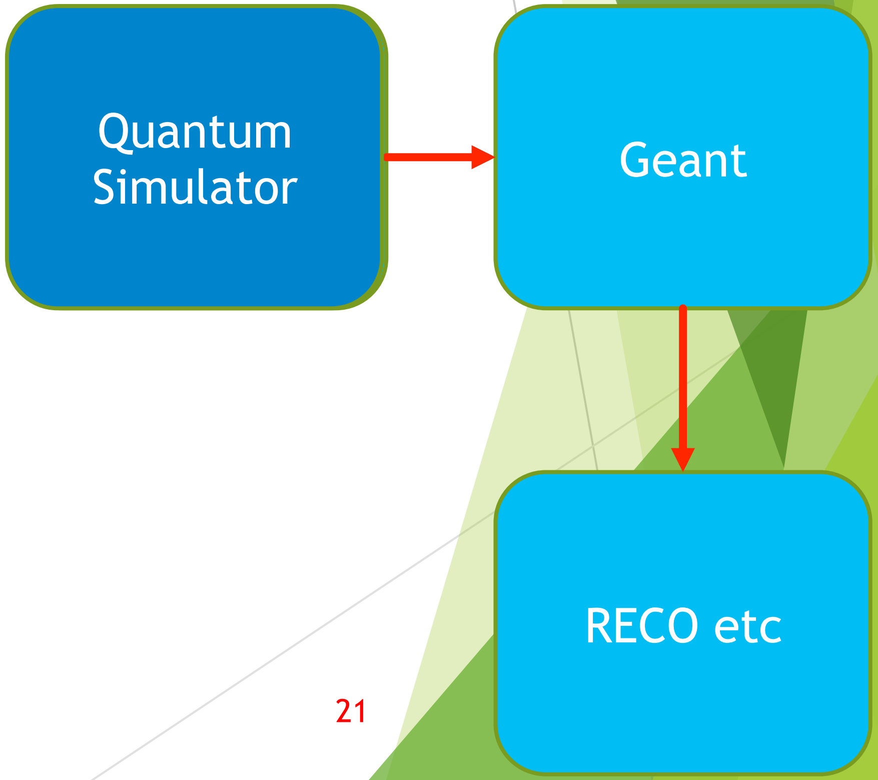
Higher level libraries are what we need, to be used as drop-in replacement in our code

- A few possible examples where QC could help (to be made more explicit in the following pages):
  1. Parton shower simulation: inputs are initial partons, outputs (semi) stable particles
  2. Finding minima as drop-in replacement to classical tools like MINUIT
     - high dmensional fuctions, binned functions, non analytical functions, …
  3. Use quantum entanglement to explore at the same time large parts of the phase space
     - «loop unrolling»



1. Quantum simulation (left to Massimo)

- Idea is
  - Build a quantum system which (at least locally) has the same H than the system you want to simulate
  - impose proper initial conditions
  - Let it evolve and measure it
- Already possible for simple systems (low dimension ising)
- In principle, one day could be able to:
  - Simulate (parts of) the standard model without LO, NLO, … approximations
  - Simulate low energy QCD showering
  - …
  - Low bar of acceptance: drop-in replacement for something we already use



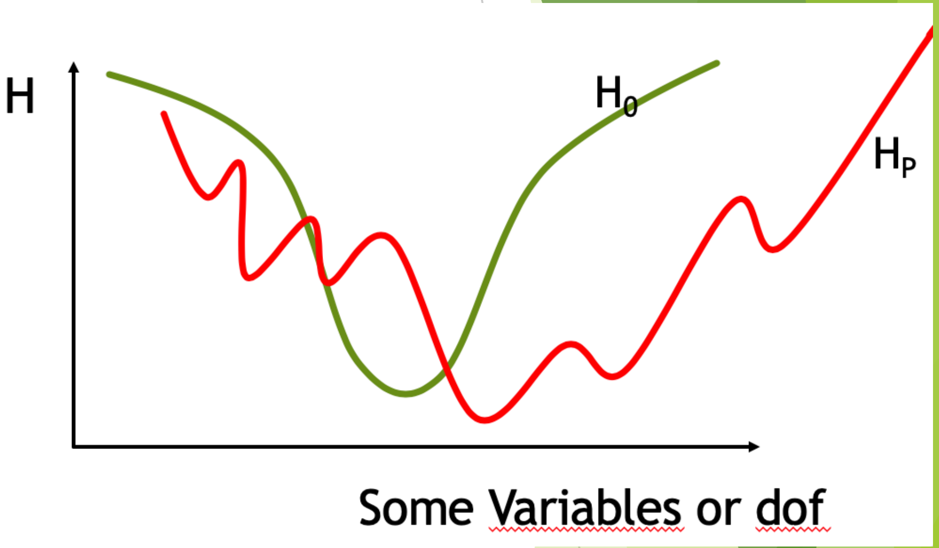
Quantum Simulator

Geant

RECO etc

2. Finding Minima

- Naive Idea is:
  - You need to minimize a ***f*(*x_1*,*x_2*,*x_3*,……*x_N*)**
  - Build a quantum system with a proper number of qubits and and hamiltonian HP~f
  - Find the ground state. It is by definition close to the minimum of f
- In practice a little more complicated. Use Adiabatic Theorem
  - Prepare a quantum system with a known behavior H_0 and put it in the lowest energy state
  - Adiabatically add «slowly» the H_P you want to minimize
  - **s=0 → s=1 «slowly»**
  - The system will find itself in the minimum for H_P

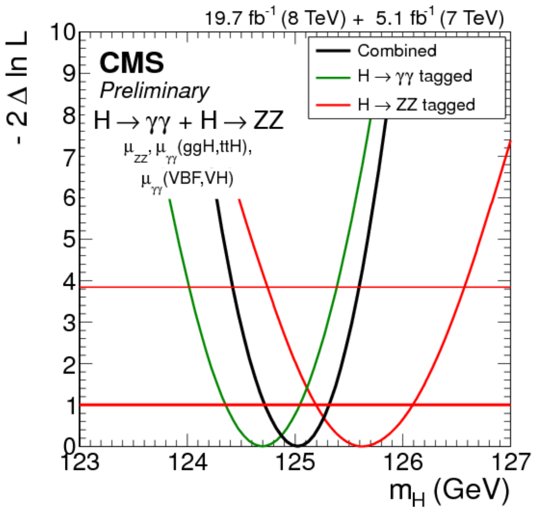


$$H(s)=(1-s)H_0+s\,H_P$$

Why is finding minima interesting?#1

- Some of today’s HEP algorithms are already expressed in terms of finding minima
  - Likelyhoods for measurement / exclusion limits
- But most of our algorithms are not:
  - Tracking: iterative
    - «given a track candidate, search for an additional measurement in an outer layer»
  - Jet finding: iterative
    - start from «seed signals», and add closeby signals until a certain category is met

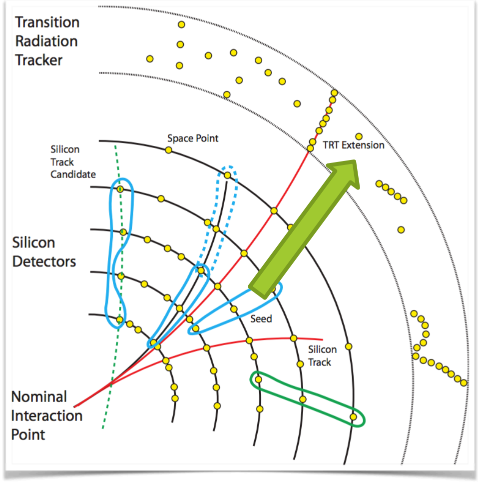
«best» Higgs mass from a likelihood fit



20

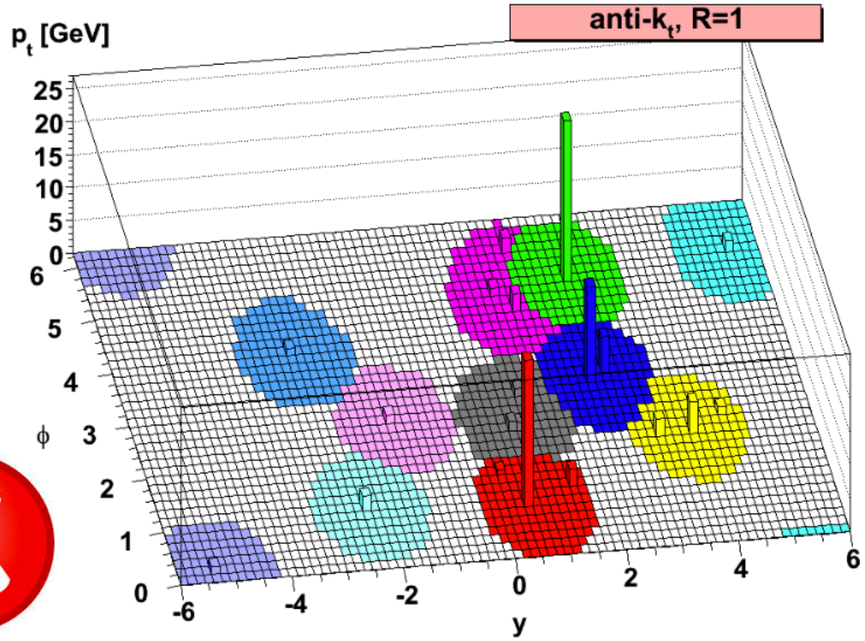
Track Reconstruction

Multi-step iterative Kalman filter approach



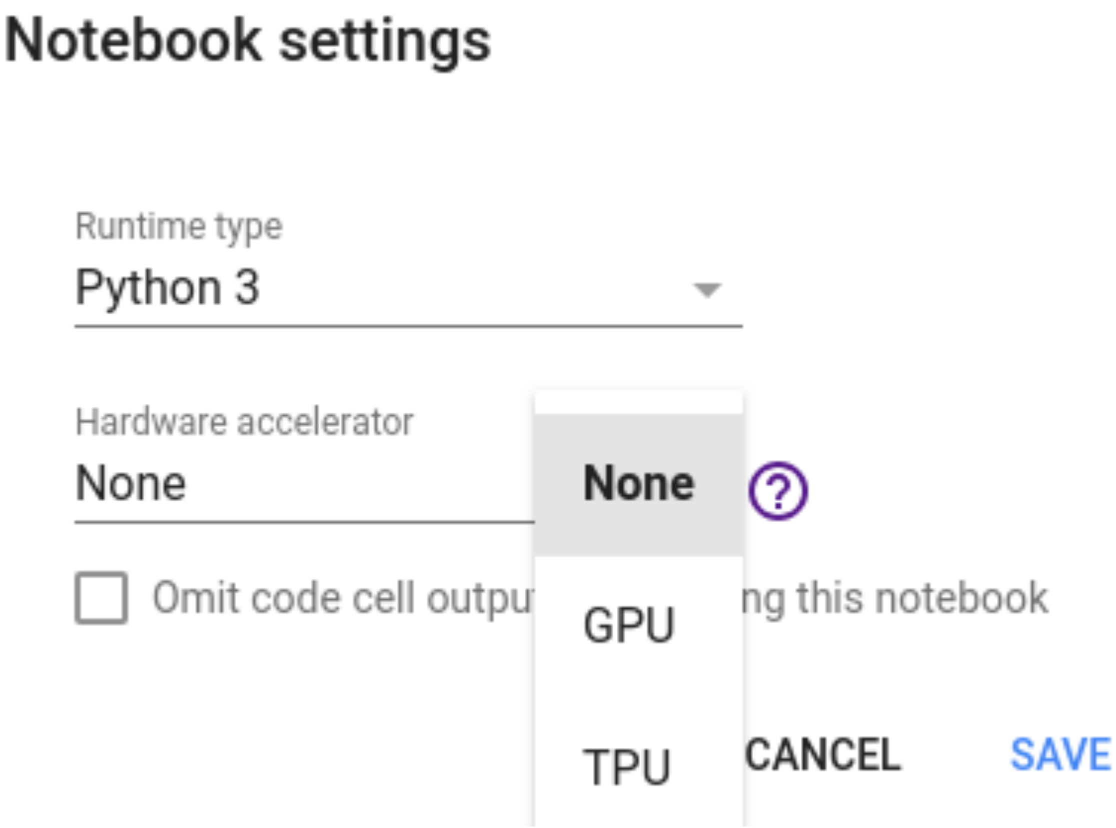
Space point formation
↓
Seed finding
↓
Track finding
↓
Ambiguity Solving
↓
TRT Extension

Jet clustering



Why is finding minima interesting?#2

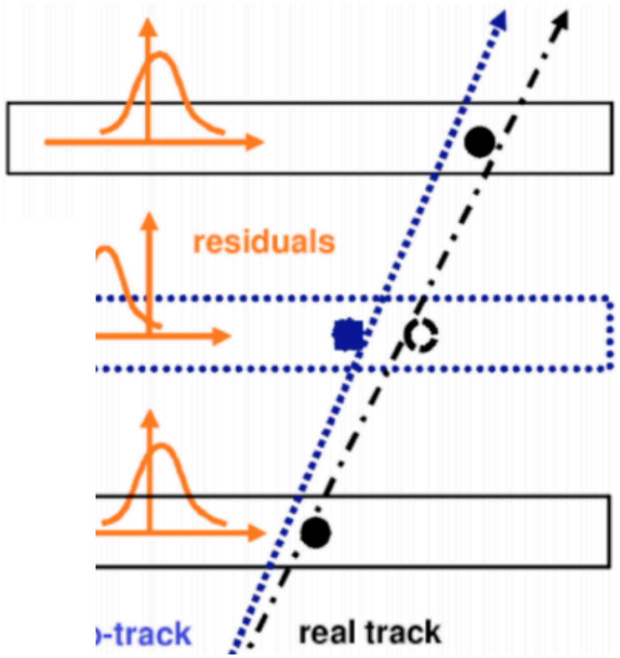
- If a fast / reliable QC minimi available, we could redesign finding minima
  - Tracking: minimize global t OR use combinatorial nergy
  - Jet finding: minimize some phi)
  - Template-driven analyses (
- And the Holy Graal: Machine minimization of some Loss Fu
  - Currently big size (future) QC cost effective?
  - Google/Cirq «promise»: rer option in Tensorflow


Notebook settings

Runtime type
Python 3

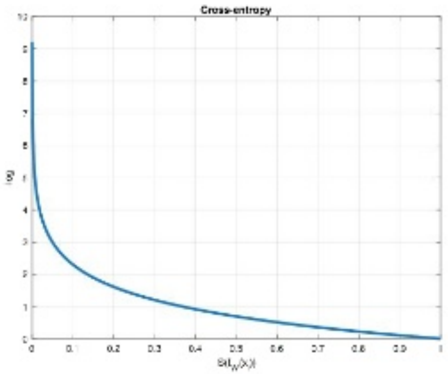
Hardware accelerator
None

☐ Omit code cell outpu ng this notebook



ntropy loss (2)

$$-\sum_k y_k \log(S(l_k)) = -\log(S(l))$$



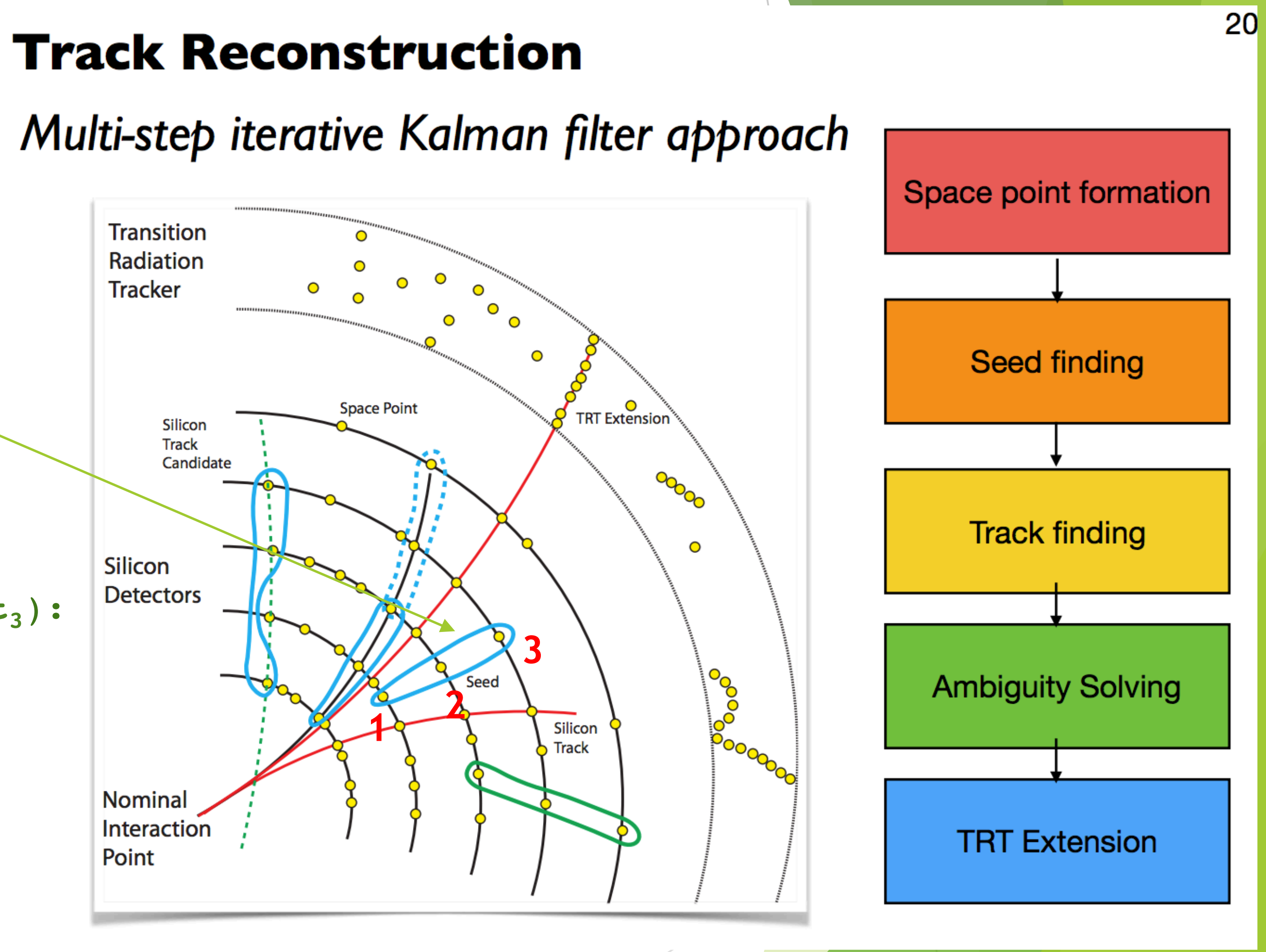
- A big advantage: no learning curve! Simply replace MINUIT / your current tool with a QC «black box». No need to understand entanglement etc

3. Loop unrolling

- Typical interesting & expensive algorithms are combinatorial in nature:
  - Seeding algorithms: find a set of n (3 here) hits compatible from being from the same track

```
for $hit_1$ in $Layer_1$:
    for $hit_2$ in $Layer_2$:
        for $hit_3$ in $Layer_3$:
            if are_compatible($hit_1$, $hit_2$, $hit_3$):
```

  - (a naive) classical implementation scales as $\mathbf{total\_hits}_1 * \mathbf{total\_hits}_2 * \mathbf{total\_hits}_3$, so essentially cubic with event complexity ~ instantaneous luminosity
- How would a possible QC algorithm be faster here?



20

Track Reconstruction

Multi-step iterative Kalman filter approach

Space point formation

Seed finding

Track finding

Ambiguity Solving

TRT Extension

QC seeding via **Grover**? (or in general how to think of a generic algorithm which selects objects among others)

- Imagine a quantum system in which eigenvectors/qubits are all the possible seeds
  - Possible seeds are thus $|0>, |1>, \ldots, |N>$ You can think at each state as a $|i> = |hit_{layer1}^{\alpha}, hit_{layer2}^{\beta}, hit_{layer3}^{\gamma}>$
  - (N ~ `total_hits`$_1$ `*` `total_hits`$_2$ `*` `total_hits`$_3$)
- Imagine a quantum unitary operator U which evaluates whether a condition is matched by a state («is it a valid seed»?)
  - $U|x> = -|x>$ if $|x>$ is a seed (it means *are_compatible*() would return True)
  - $U|x> = |x>$ if $|x>$ is not a seed (it means *are_compatible*() would return False)
  - U can be thought as $U = (-1)^{are\_compatible(x)}$
- Prepare a uniform initial state $|s_0> = 1/\sqrt{N} \sum_{0}^{n} |i>$ (all possible seeds equiprobable)
- What happens to $|s_0>$ when passing through U? Let’s assume there is only one valid seed $|j>$
  - $|s_0> \xrightarrow{U} |s_{\perp}> := 1/\sqrt{N} \sum_{i \neq j} |i> - 1/\sqrt{N}\ |j>$
- Define a new operator which «flips» the state with respect to previous $|s>$ (call it U_f)
- The subsequential application of UU_f **O(sqrt(N))** amplificates the amplitude of $|j>$ and reduces all the others

$$UU_f \ldots UU_f\ |s_0> \rightarrow |j>$$

Where is the trick (why cannot such a thing be used without QC)?

- Standard computer: when evaluating $U|s>$ the only real way is to loop on i, which indeed means the triple nested loop. Time is ~O((total_hits$_1$ * total_hits$_2$ * total_hits$_3$))=~O(N_{hits}^3)
- A QC can use the superposition to apply the U operator on all the eigenvectors at the same time. Potentially time is O(1 cycle), if we apply the M times it is O(M) (M ~ sqrt(N)~sqrt(N_{hits}^3))
- This is theoretically valid in general, if you are able to
  - Have a system with enough qubits to describe the base of eigenvectors
  - You can build the operator U
  - You can maintain the entanglement long enough (and clean enough) to measure it
  - You can reproduce the setup enough times

Who can provide us with these «basic» tools?

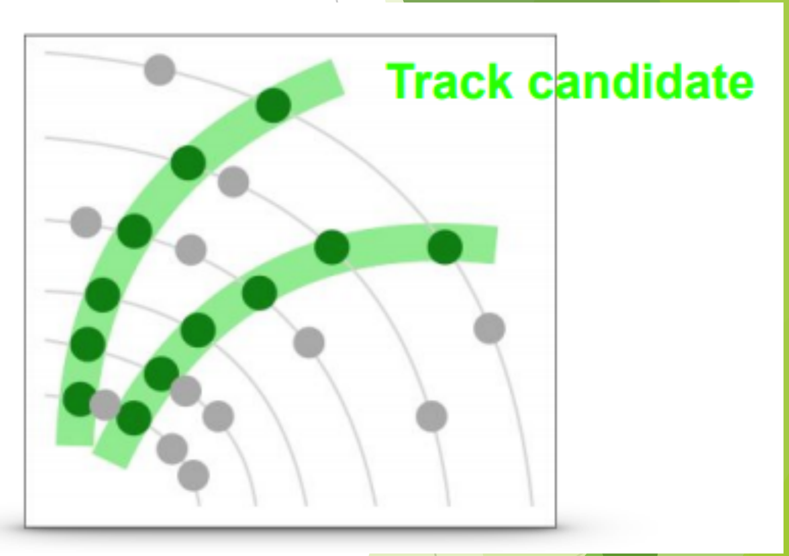
- We need close cooperation between experts of the field and the HEP people working on algorithms
- We want to be guinea pigs, and we can afford it
  - CMS Computing, as an example, has O(200) collaborators
  - Even without new manpower, it would be easy to find a few persons interested
- What do we gain?
  - Mostly, the capability to be «ready to react» in case there is a technological breakthrough
  - Being ready needs, as explained:
    - Access to systems (emulators / real)
    - Having algorithms we could use

Some HEP-ex papers

- Some references / fast examples of existing stuff
- **Spoiler #1**: nothing really usable today, but ”whenever we get enough qubits….”
- **Spoiler #2**: you will see all the examples use D-Wave hardware. It seems that today it is the only reasonable choice due to the very small size of the rest

QUBO: using quantum annealing for pattern recognition

- Given a typical silicon pixel detector and its Hits {H}, build an "energy" function which is at the minimum when {hits, doublets, triplets} beloging to the same track are considered
- In the end it is a categorization problem: you list all the possible inputs, you match it to a qubit, and in the end ("measurement") the qubit will collapse to 0 or 1.
- Currenlty not easily doable: if 5000 hits overall
  - O(5000^2) doublets → QUBO starting from (preselected) doublets
  - O(5000^3) triplets → QUBO starting from (preselected) triplets
  - D-wave: ~1000 not fully connected qubits (said to be equivaelnt to ~30 ideal qubits)
- QUBO: quadratic unconstrained binary optimization



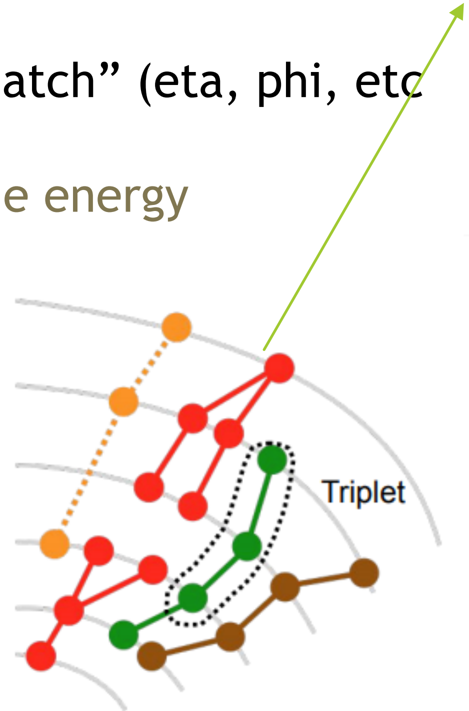
If triplets …

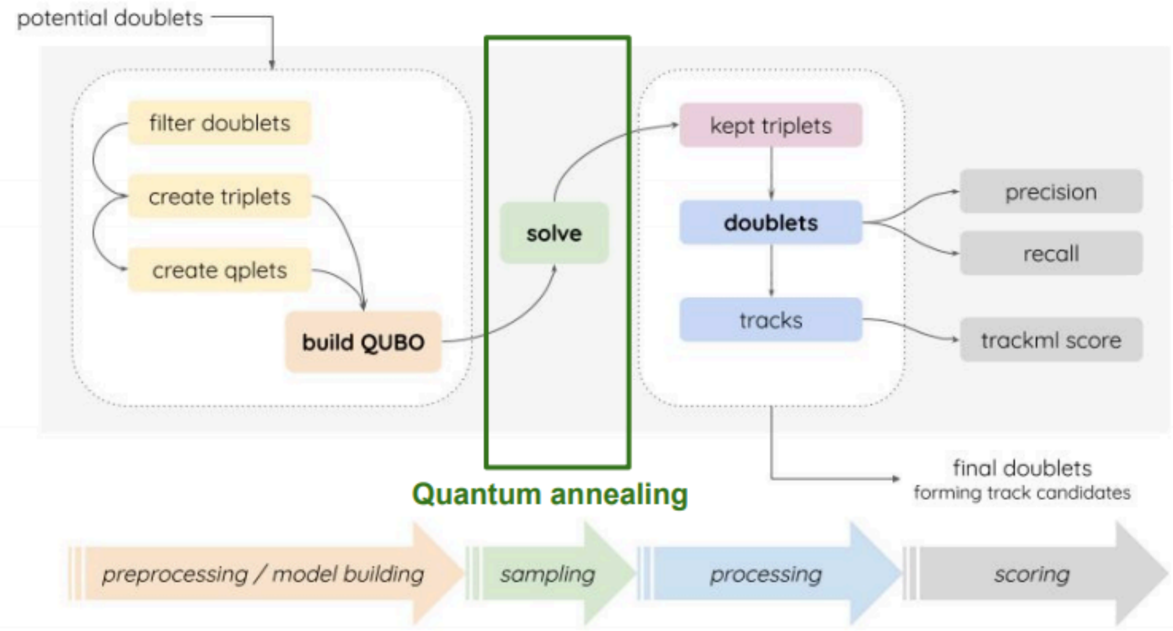
$$E = \alpha\left(\sum_{i}^{N} T_i\right) \boxed{-\left(\sum_{i,j} S_{ij}T_iT_j\right)} \boxed{+\zeta\left(\sum_{i,j} T_iT_j\right)}, \quad T \in \{0,1\}$$

qubit

bias weight Connection strength Avoid conflicts, zigzag pattern, holes

- S is larger is the two triplets “match” (eta, phi, etc phase space)
  - A good matching reduces the energy
- The other term avoid conflicts
  - They increase the energy
- Holes can be allowed





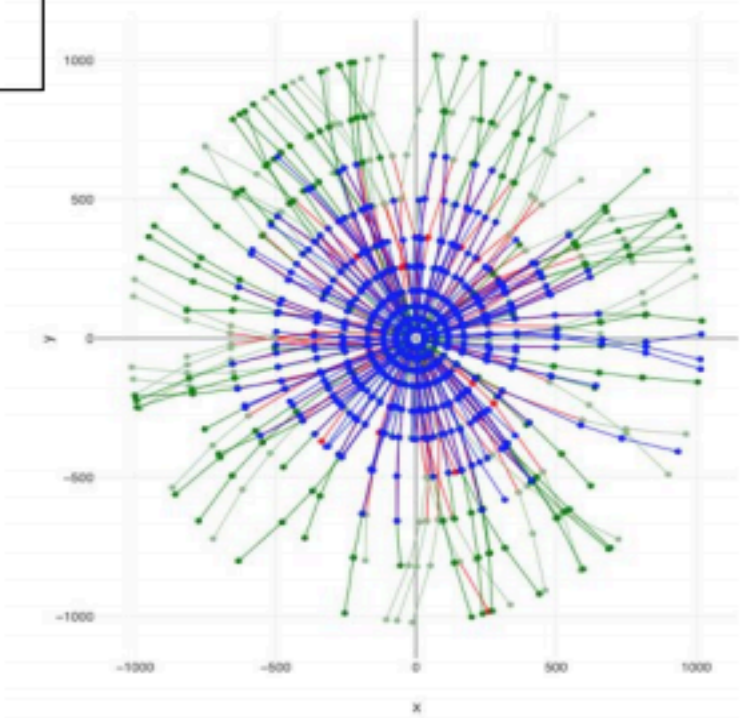
If triplets …

Results

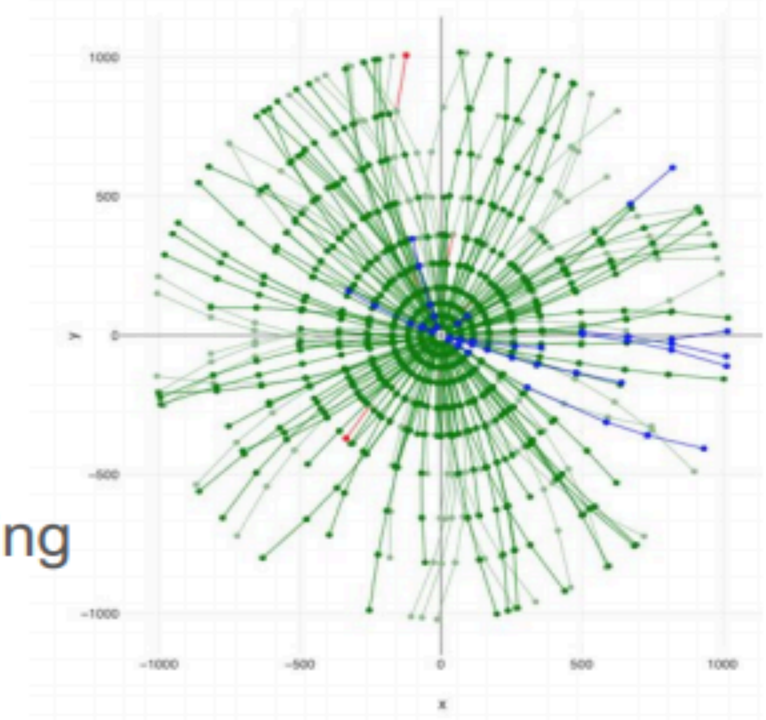
1600 particles (20% of HL-LHC)
- 11000 hits

🟩 Reconstructed high pT tracks
🟩 Reconstructed low pT tracks
🟦 Not reconstructed tracks
🟥 Fake tracks

Input → Doublet selection



→ Annealing



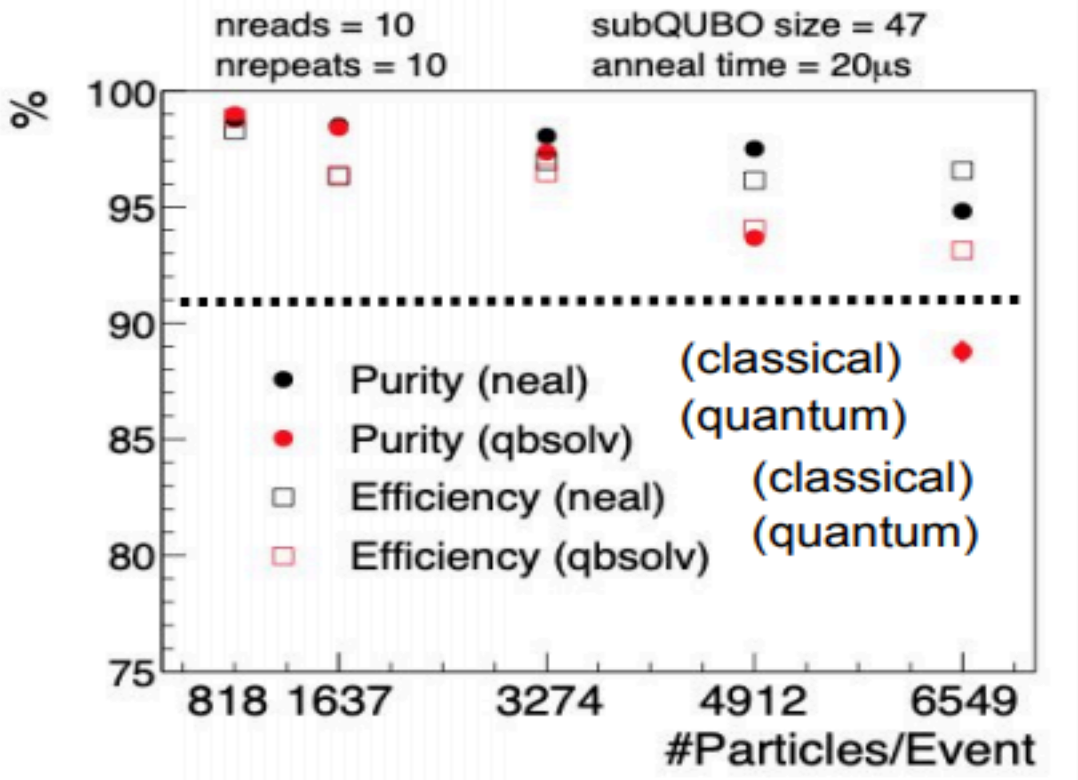
390000 Doublets	
Purity	0.22 %
Efficiency	99.5 %

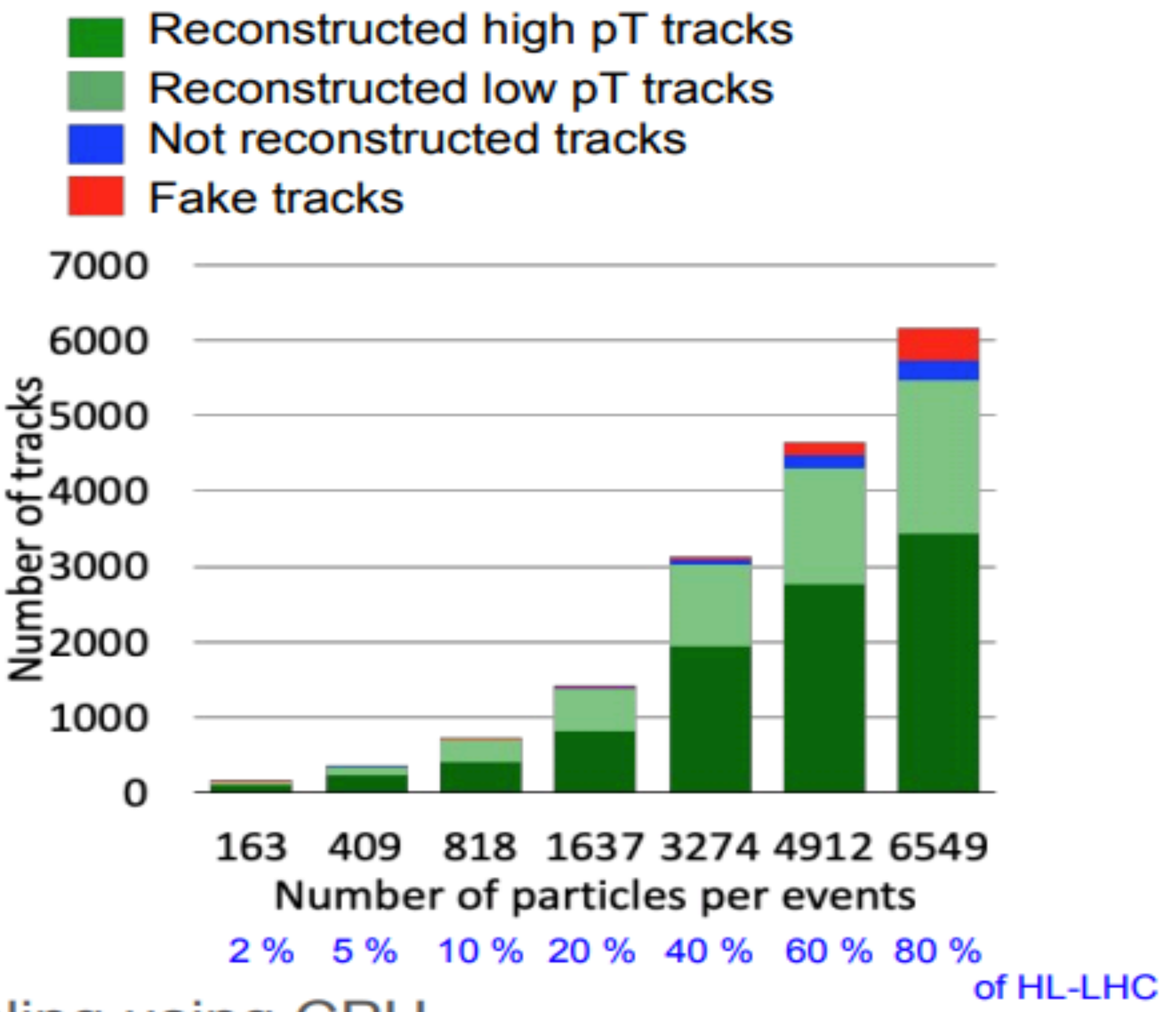
2445 Doublets

1424 Doublets	
Purity	98.5 %
Efficiency	96.4 %

If triplets …

Results

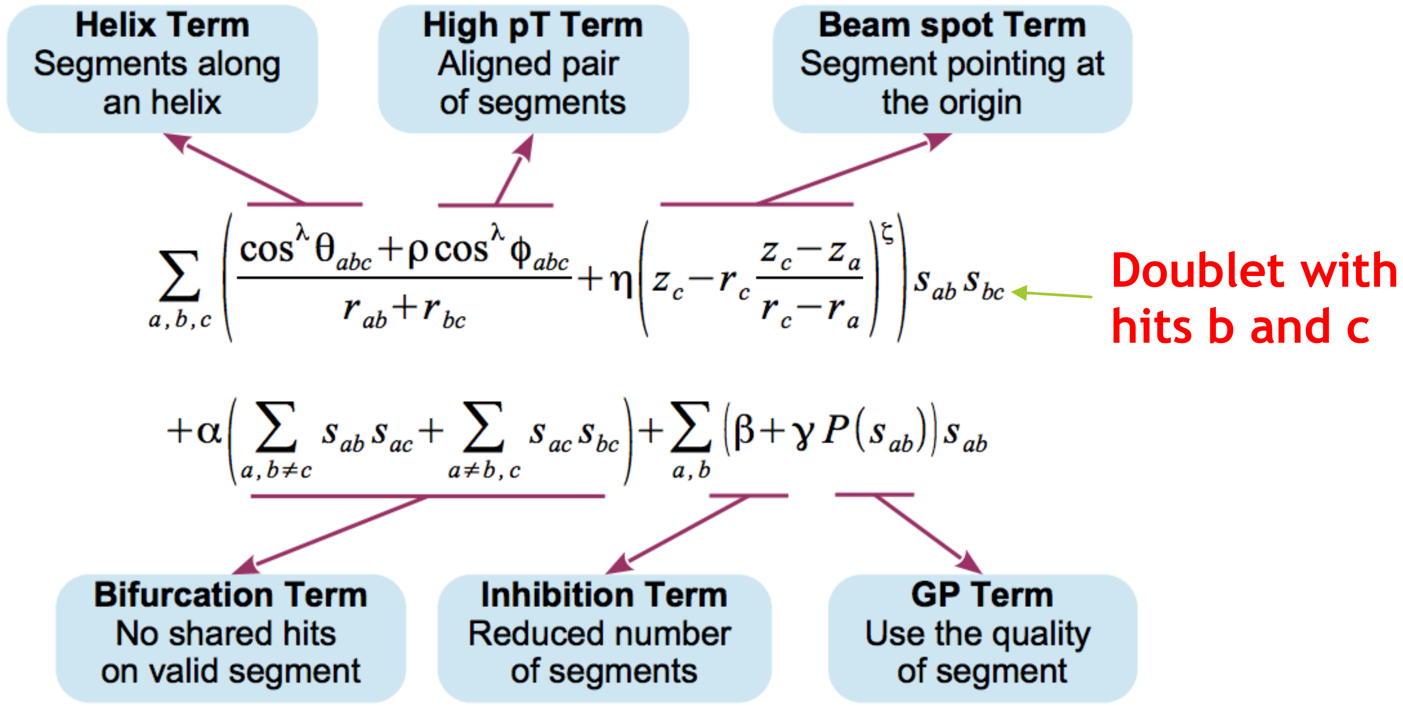




- Reference solver: neal = simulated annealing using CPU
- >90 % efficiency / purity below 6000 particles environment
- Equivalent performance with the classical annealing (neal)

If doublets …

- Larger input set, you need more qubits OR more preselection OR partitioning the problem (eta, phi slices, …)



Helix Term
Segments along an helix

High pT Term
Aligned pair of segments

Beam spot Term
Segment pointing at the origin

$$\sum_{a,b,c}\left(\frac{\cos^{\lambda}\theta_{abc}+\rho\cos^{\lambda}\phi_{abc}}{r_{ab}+r_{bc}}+\eta\left(z_c-r_c\frac{z_c-z_a}{r_c-r_a}\right)^{\xi}\right)s_{ab}s_{bc}$$

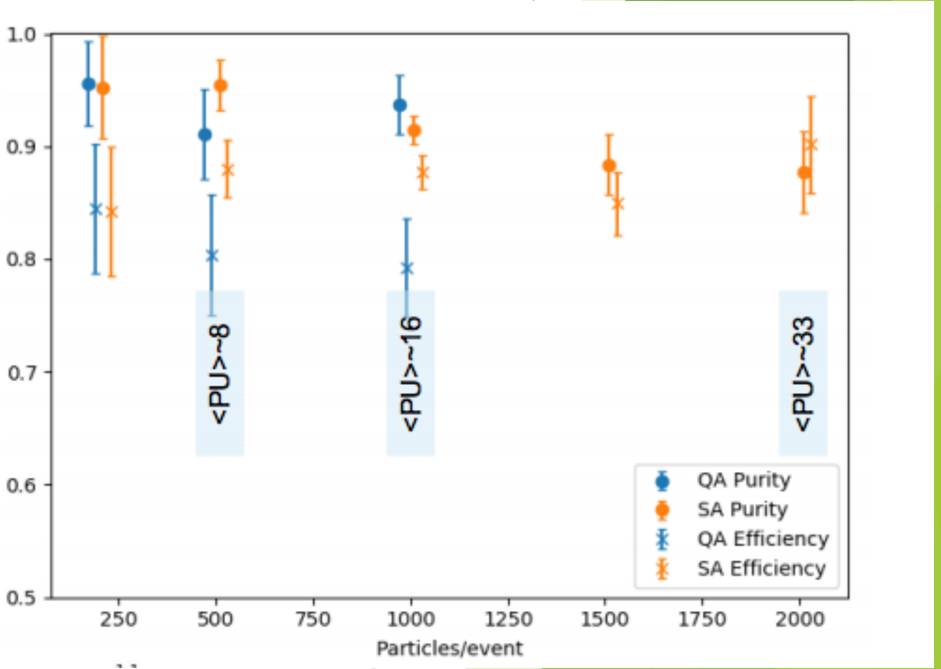
Doublet with hits b and c

$$+\alpha\left(\sum_{a,b\neq c}s_{ab}s_{ac}+\sum_{a\neq b,c}s_{ac}s_{bc}\right)+\sum_{a,b}\left(\beta+\gamma P(s_{ab})\right)s_{ab}$$

Bifurcation Term
No shared hits on valid segment

Inhibition Term
Reduced number of segments

GP Term
Use the quality of segment



Can compared Quantum Annealing (QA) with Simulated annealing (SA) only up to current PU ~ 20; need larger systems

Current performance…

- Only limited datasets can be used on the D-Wave Quantum Annealer; still:
  - Annealing time ~ 0.5 sec (triplets)
  - Preprocessing time (building the triplets) 4 sec
- A clear example of the bottlenecks we can find using QC: the “core algorithm” can be fast, but the preparation + the data movement can be problematic
- Same in Grover shown before: how much time is hidden behind “prepare all the possible seeds”?

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

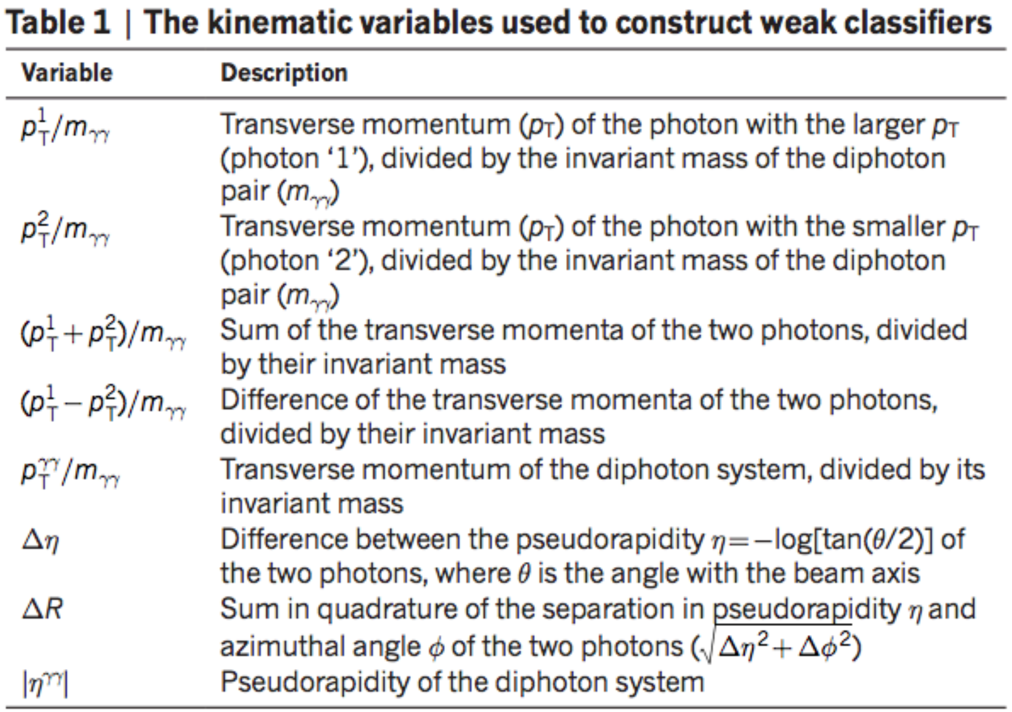
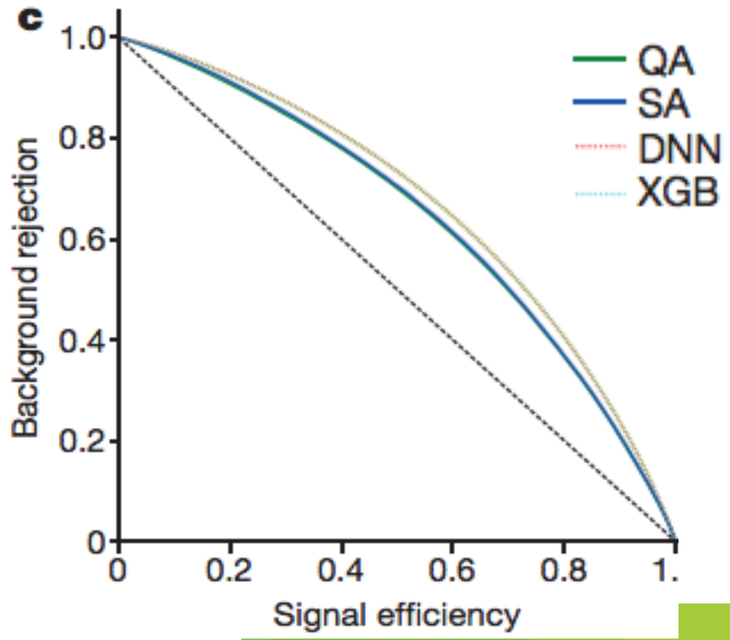


Table 1 | The kinematic variables used to construct weak classifiers

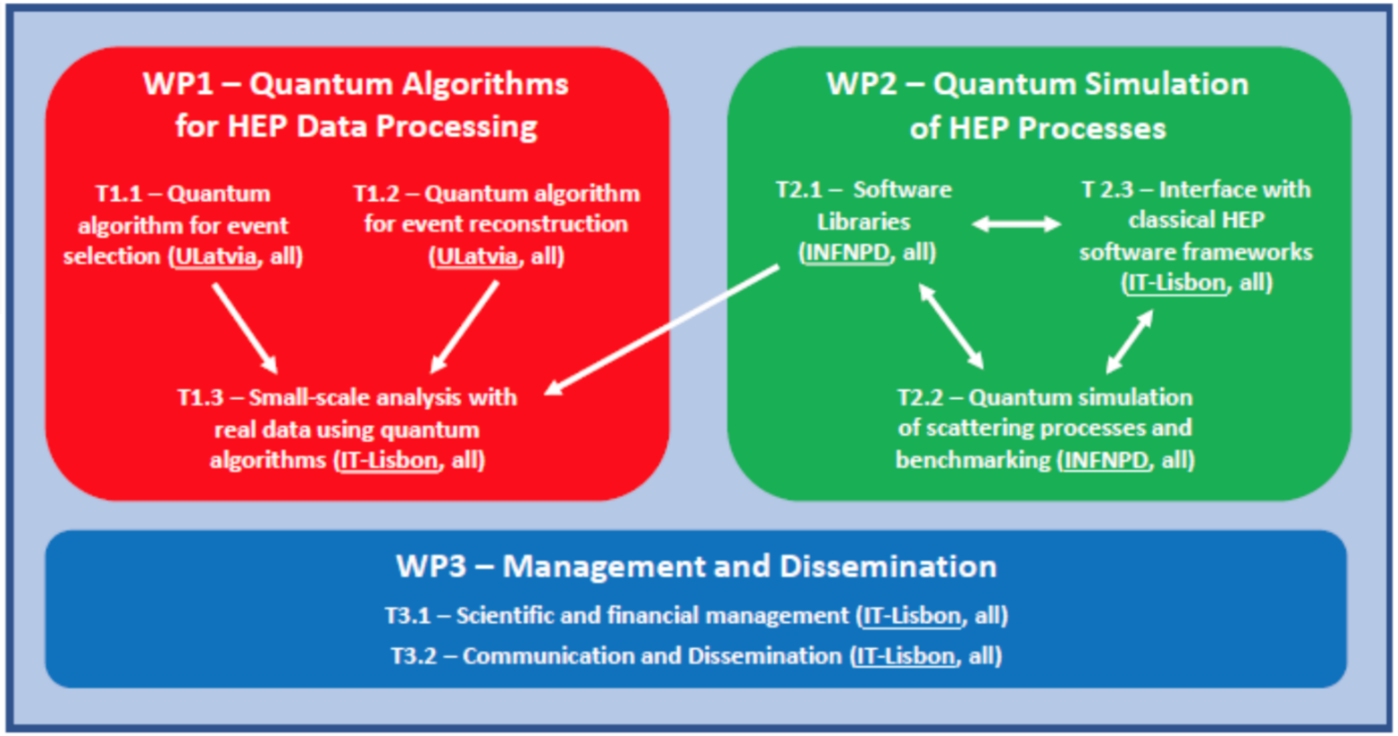
Variable	Description
$p_T^1/m_{\gamma\gamma}$	Transverse momentum (p_T) of the photon with the larger p_T (photon ‘1’), divided by the invariant mass of the diphoton pair ($m_{\gamma\gamma}$)
$p_T^2/m_{\gamma\gamma}$	Transverse momentum (p_T) of the photon with the smaller p_T (photon ‘2’), divided by 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 ($\sqrt{\Delta\eta^2+\Delta\phi^2}$)
$\lvert\eta^{\gamma\gamma}\rvert$	Pseudorapidity of the diphoton system



QuantHEP

Quantum Computing Solutions for High-Energy Physics

- Successful application to a QuantERA call
- INFN (PD), LIP, ULatvia
- Main tasks:
  1. Develop quantum algorithms for event selection and event reconstruction.
  2. Develop the quantum simulation of scattering processes.
  3. Benchmark the performance of our quantum solutions against small-sets of simulated and real data from CERN.



WP1 – Quantum Algorithms for HEP Data Processing

T1.1 – Quantum algorithm for event selection (ULatvia, all)

T1.2 – Quantum algorithm for event reconstruction (ULatvia, all)

T1.3 – Small-scale analysis with real data using quantum algorithms (IT-Lisbon, all)

WP2 – Quantum Simulation of HEP Processes

T2.1 – Software Libraries (INFNPD, all)

T 2.3 – Interface with classical HEP software frameworks (IT-Lisbon, all)

T2.2 – Quantum simulation of scattering processes and benchmarking (INFNPD, all)

WP3 – Management and Dissemination

T3.1 – Scientific and financial management (IT-Lisbon, all)

T3.2 – Communication and Dissemination (IT-Lisbon, all)

For Exp-HEP, this is the important part, and reflects somehow the previous discussion:

- Provide drop-in high(er) level libraries to physicists
- Start with small benchmarks, and scale when hardware available

Additional notable (for HEP) initiatives

- CERN, via OpenLab, is launching collaborations with at least IBM, Google and D-Wave to have access to the real machines
- US has funded QC researches at FNAL and LBNL
- Germany has put 650MEur on Quantum Computing
  - DESY and Fraunhofer should get a machine each (IBM and D-Wave?)
- Europe has put 1BEur on the Quantum Flagship (soon 3B?)
- INFN has entered the Quantera Consortium
  - Funding opportunities
- ATLAS, LHCb and CMS (with CERN/DR endorsement) have submitted a Training Network proposal for HL-LHC Software and Computing
  - "IFRIT": Implementing the Future computing Roadmap In Training
  - If successful, Pisa has in the project 1 Early Stage Researcher (partially on QC) to be funded in late 2020; in the project also the Quantum Labs of IBM/Zurich
    - In the plans: write high level basic libraries for HEP

Overall

- A reasonable approach for us (== LHC experiment, but in general HEP-ex) seems to be
  - We honestly **do not think** we can count on QS/QC as a **mission critical tool** for HL-LHC …
  - … but equally, we **cannot be caught unprepared** in the eventuality of a technology / theory breakthrough
- We are sure we can find in our Collaborations interest in following / experimenting / studying QS/QC matters
  - If we are given some initial **guidance**
  - If we are given **access** to emulators / real systems
  - You are seeing today some examples of such activities by single / small groups
  - **Which is the best way to scale activities?**