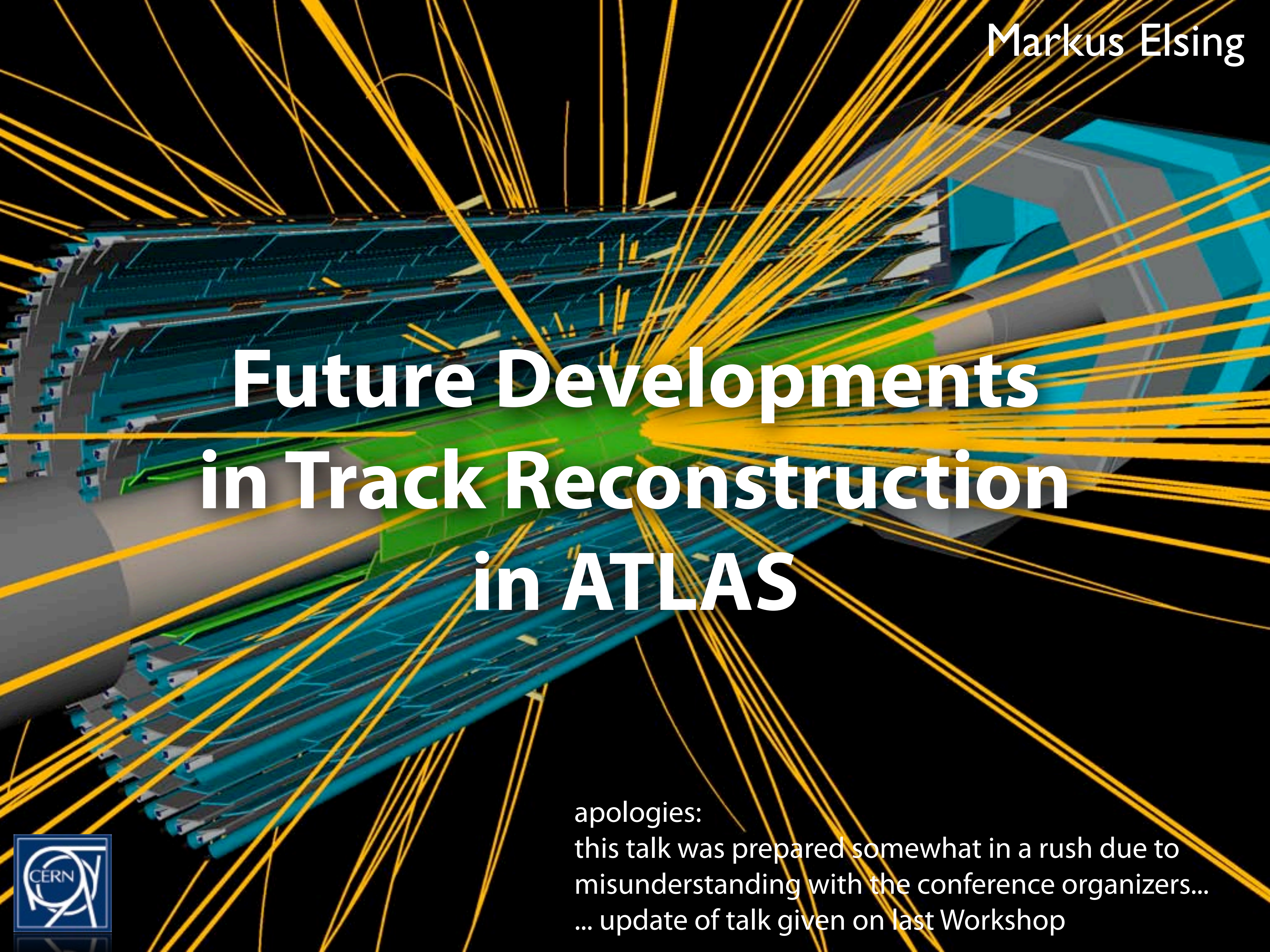


Markus Elsing

Future Developments in Track Reconstruction in ATLAS

apologies:
this talk was prepared somewhat in a rush due to misunderstanding with the conference organizers...
... update of talk given on last Workshop

Outline

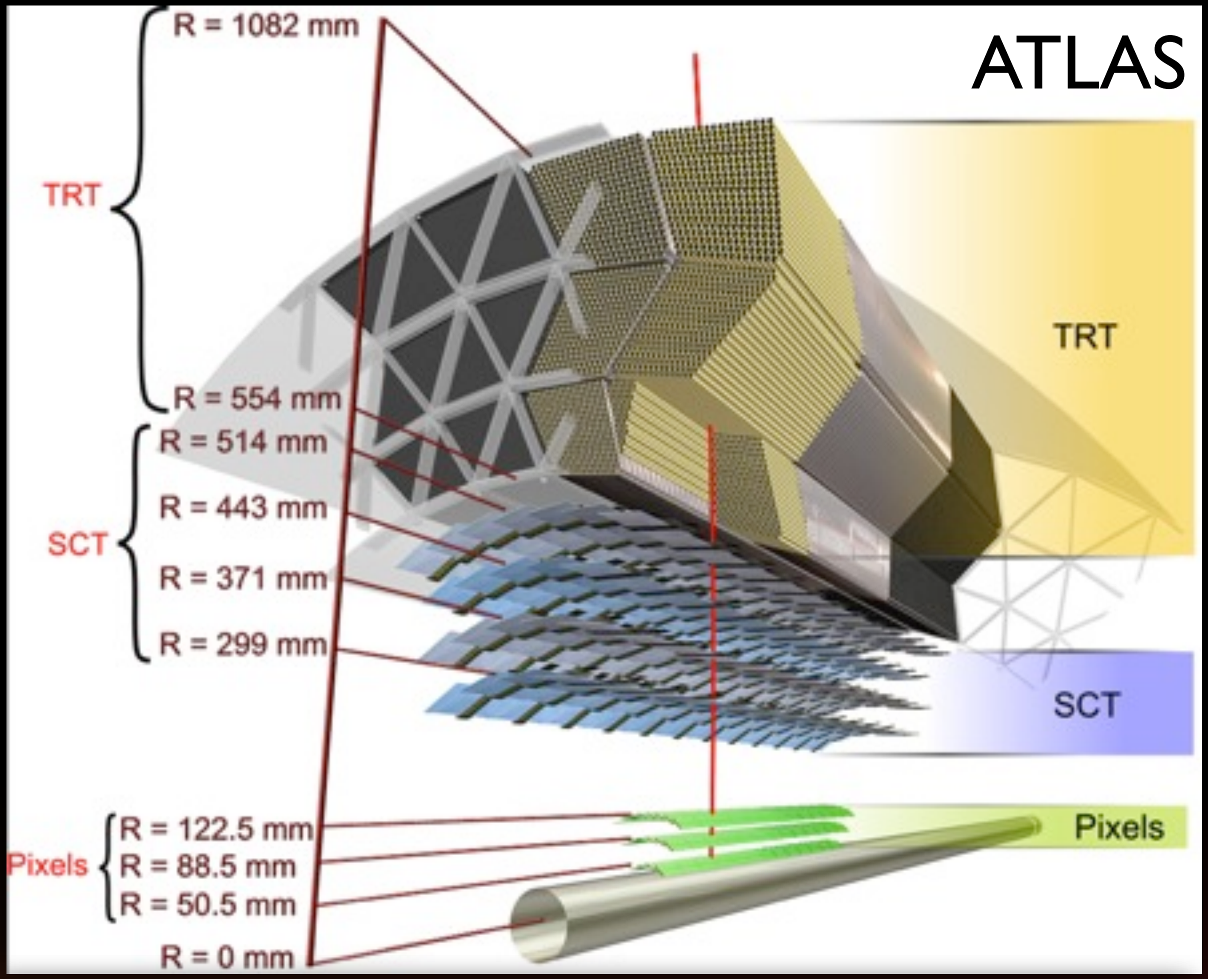
- short introduction
  - ➡ motivation, present technology and recent improvements
- ATLAS upgrade program
  - ➡ summary of Inner Detector updates
- CPU performance vs Pileup
  - ➡ tracking software development program to tackle the CPU limitations
- few words on (fast track) simulation
- trigger upgrade and tracking

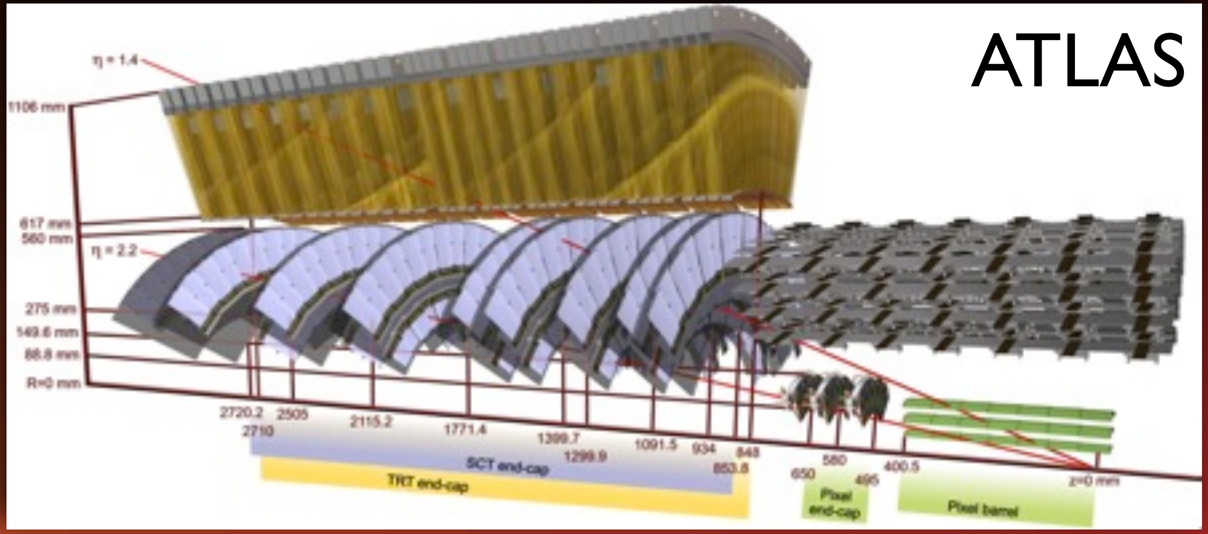
Introduction

- requirements on ATLAS Inner Detector
  - ➡ **precision tracking** at LHC luminosities (central heavy ion event multiplicities) with a hermitic detector covering 5 units in η
  - ➡ precise **primary/secondary vertex** reconstruction and to provide excellent **b-tagging in jets**
  - ➡ reconstruction of **electrons** (and converted photons)
  - ➡ tracking of **muons** combined with muon spectrometer, good resolution over the full accessible momentum range
  - ➡ enable (hadronic) **tau**, exclusive **b-** and **c-hadron** reconstruction
  - ➡ provide **particle identification**
    - transition radiation in ATLAS TRT for **electron identification**
    - as well dE/dx in Pixels or TRT
  - ➡ not to forget: enable fast tracking for **(high level) trigger**
- constraints on detector design
  - ➡ **minimize material** for best precision and to minimize interactions before the calorimeter
  - ➡ increasing **sensor granularity** to reduce occupancy
    - increase number of electronics channels and heat load
    - leading to more material

ATLAS Inner Detector Layout

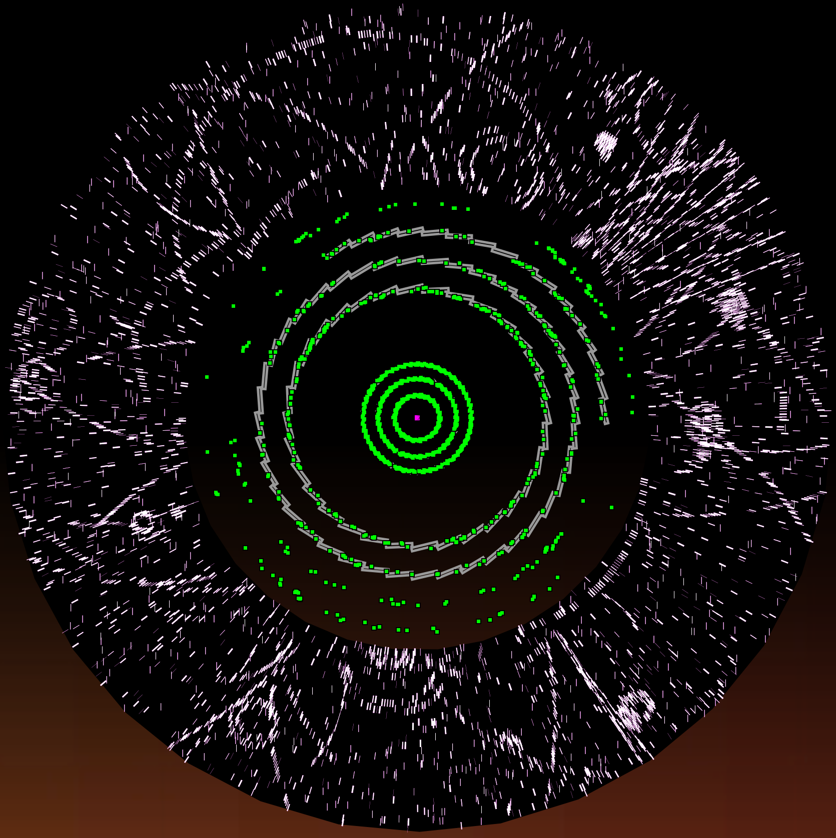
- 3 subsystems:
  - ➡ 3 layer **Pixel** system, 3 endcap disks
    - 1744 Pixel modules
    - 80.4 million channels
    - pitch 50 µm × 400 µm
    - total of 1.8 m^2
  - ➡ 4 layers of small angle stereo strips, 9 endcap disks each side (**SCT**)
    - 4088 double sided modules
    - 6.3 million channels
    - pitch 80 µm, 40 mrad stereo angle
    - total of 60 m^2
  - ➡ Transition Radiation Tracker (**TRT**)
    - typically 36 hits per track
    - transition radiation to identify electrons
    - total of 350K channels





ATLAS Track Reconstruction Chain

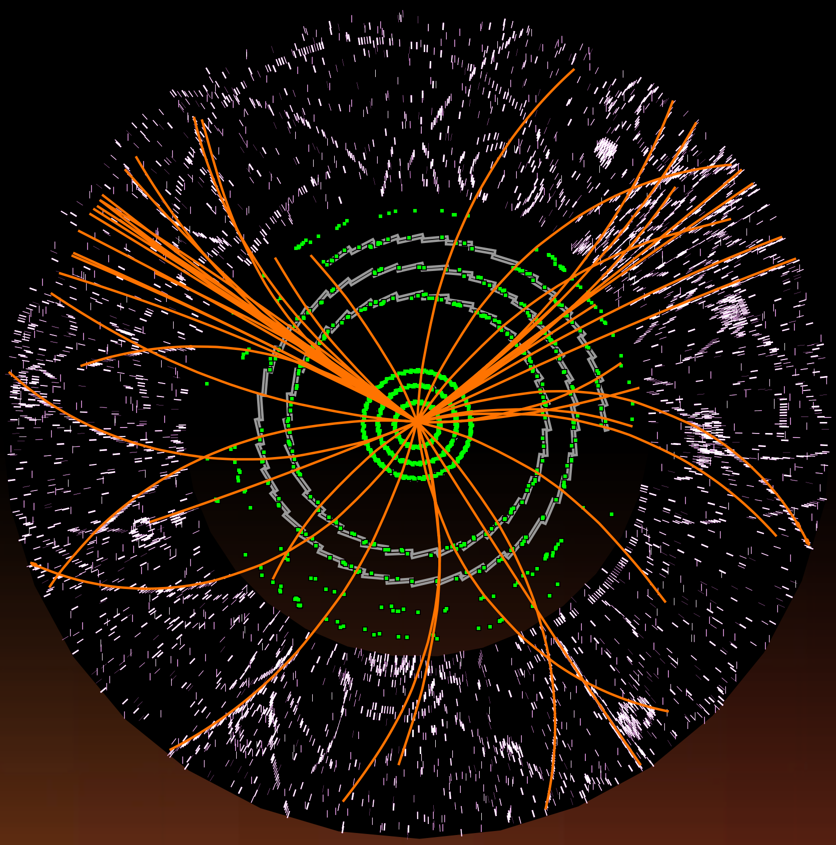
pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation



ATLAS Track Reconstruction Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates



ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

ATLAS Track Reconstruction Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

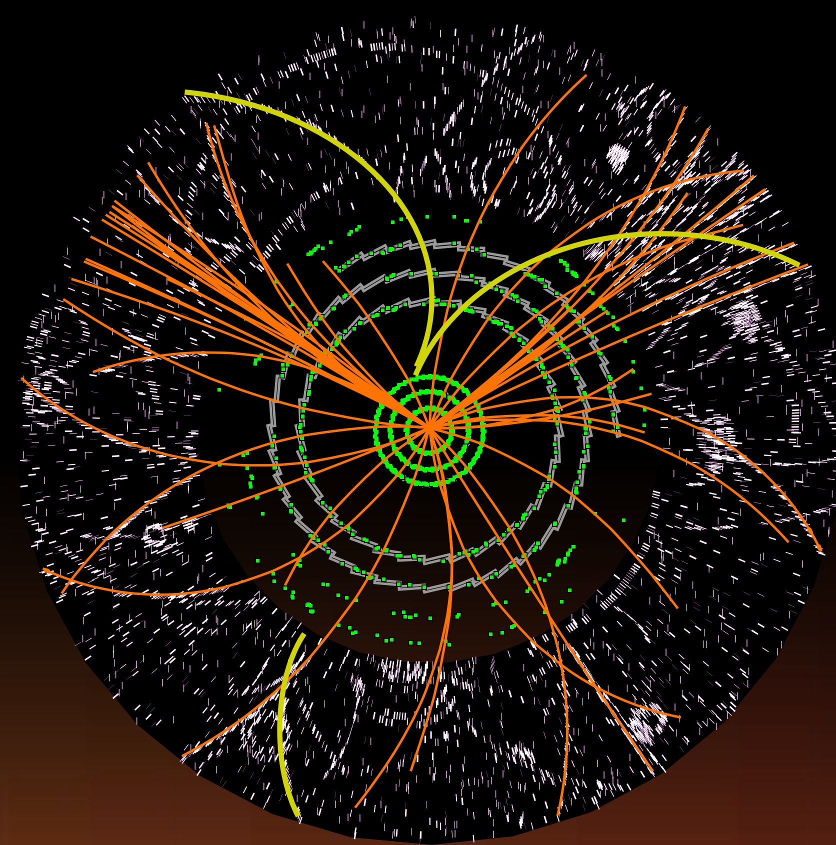
extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

standalone TRT
- ➡ unused TRT segments



ATLAS Track Reconstruction Chain

pre-precessing
- ➡ Pixel+SCT clustering
- ➡ TRT drift circle formation
- ➡ space points formation

combinatorial track finder
- ➡ iterative :
  1. Pixel seeds
  2. Pixel+SCT seeds
  3. SCT seeds
- ➡ restricted to roads
- ➡ bookkeeping to avoid duplicate candidates

ambiguity solution
- ➡ precise least square fit with full geometry
- ➡ selection of best silicon tracks using:
  1. hit content, holes
  2. number of shared hits
  3. fit quality...

extension into TRT
- ➡ progressive finder
- ➡ refit of track and selection

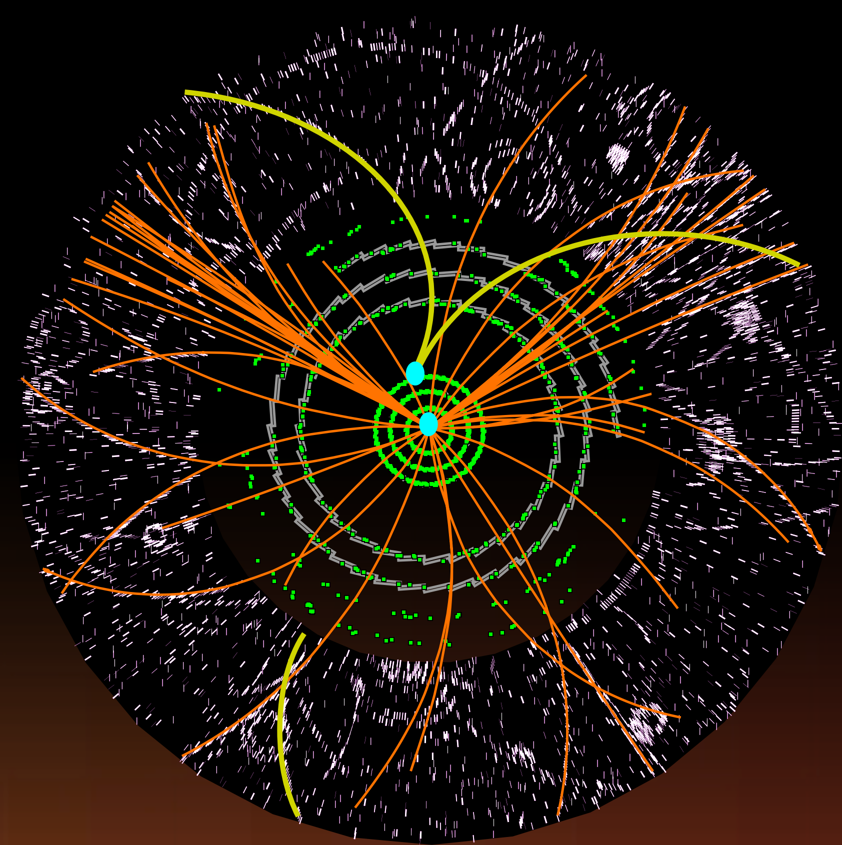
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

TRT seeded finder
- ➡ from TRT into SCT+Pixels
- ➡ combinatorial finder

ambiguity solution
- ➡ precise fit and selection
- ➡ TRT seeded tracks

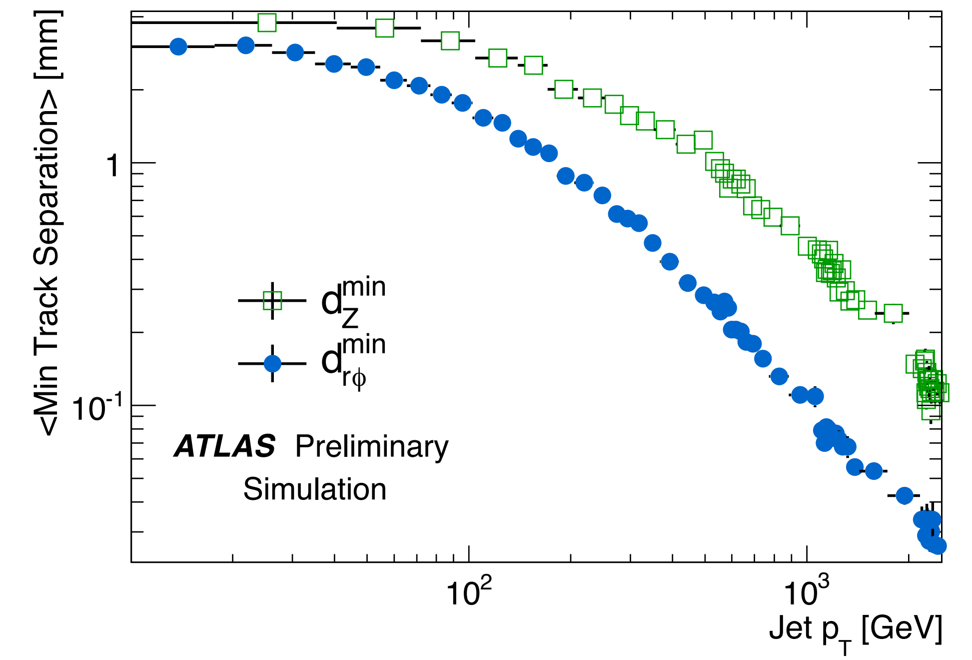
standalone TRT
- ➡ unused TRT segments

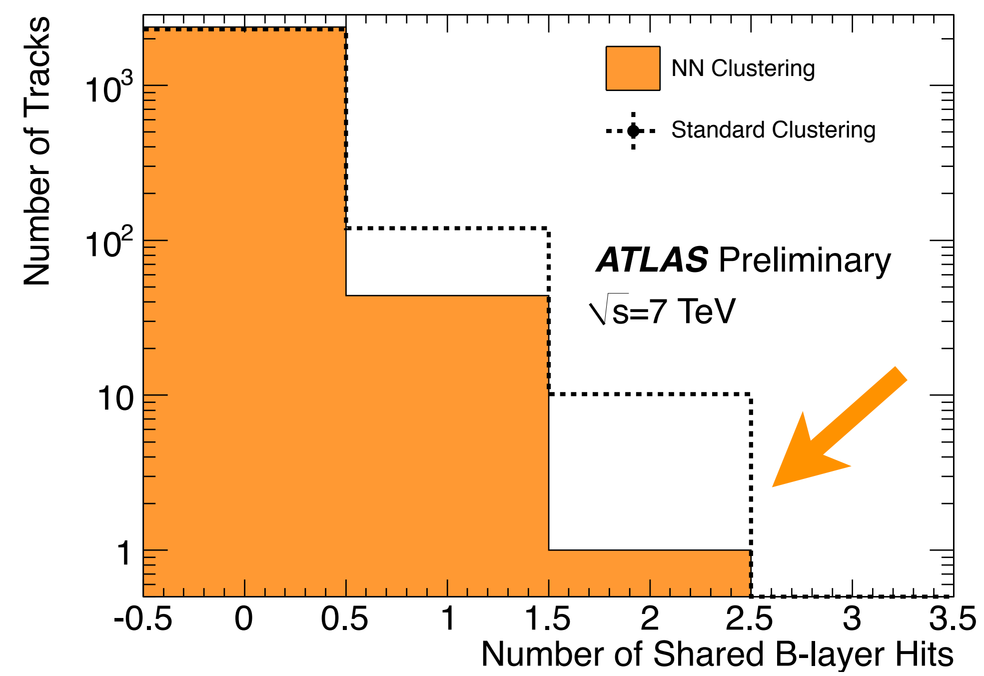
vertexing
- ➡ primary vertexing
- ➡ conversion and V0 search

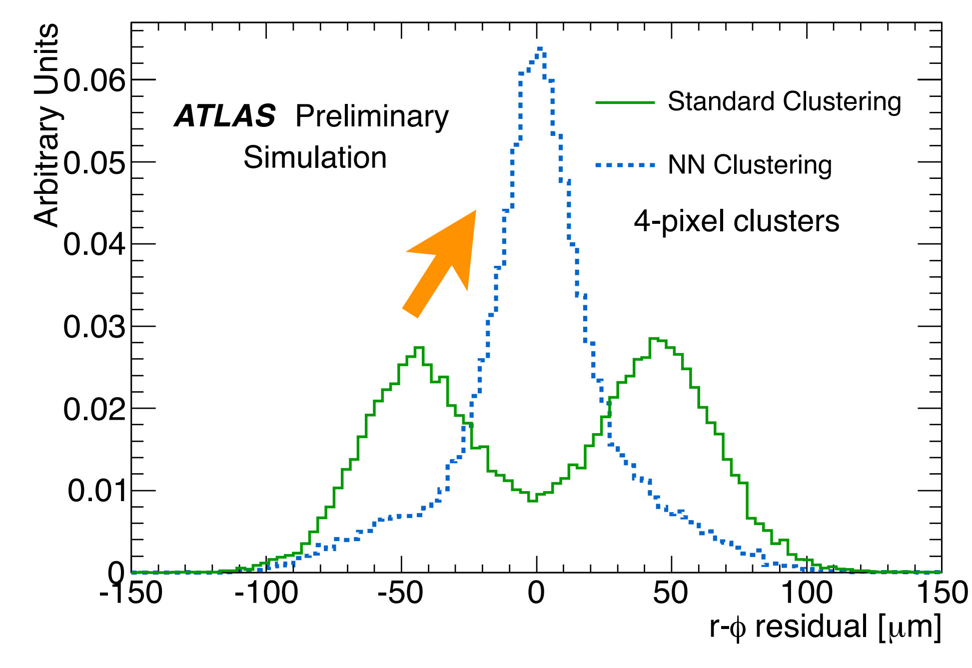


Neural Net Pixel Clustering

- novel technique, motivation:
  - ➡ high track density in jets leads to cluster merging
  - ➡ limits tracking in jets and b-tagging performance
- algorithm to split merge clusters
  - ➡ neural network (NN) based technique
    - explores analog Pixel information
  - ➡ run 5 networks:
    - NN1: probability a cluster is 1/2/>2 tracks
    - NN2: best position for each (sub)cluster
    - NN3: error estimate for cluster
    - NN4+5: redo NN2+3 using track prediction
  - ➡ adapt pattern recognition
- performance improvements
  - ➡ improved cluster resolution
  - ➡ dramatic reduction in rate of shared B-layer hits and therefore improved tracking in core of jets

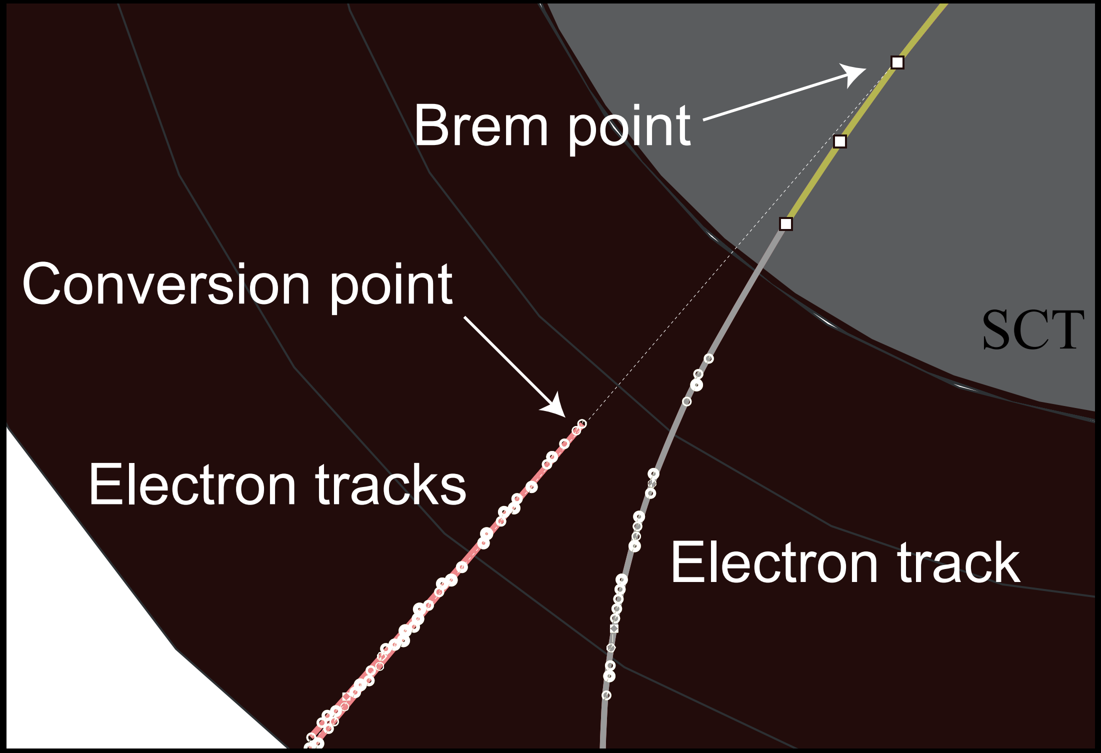


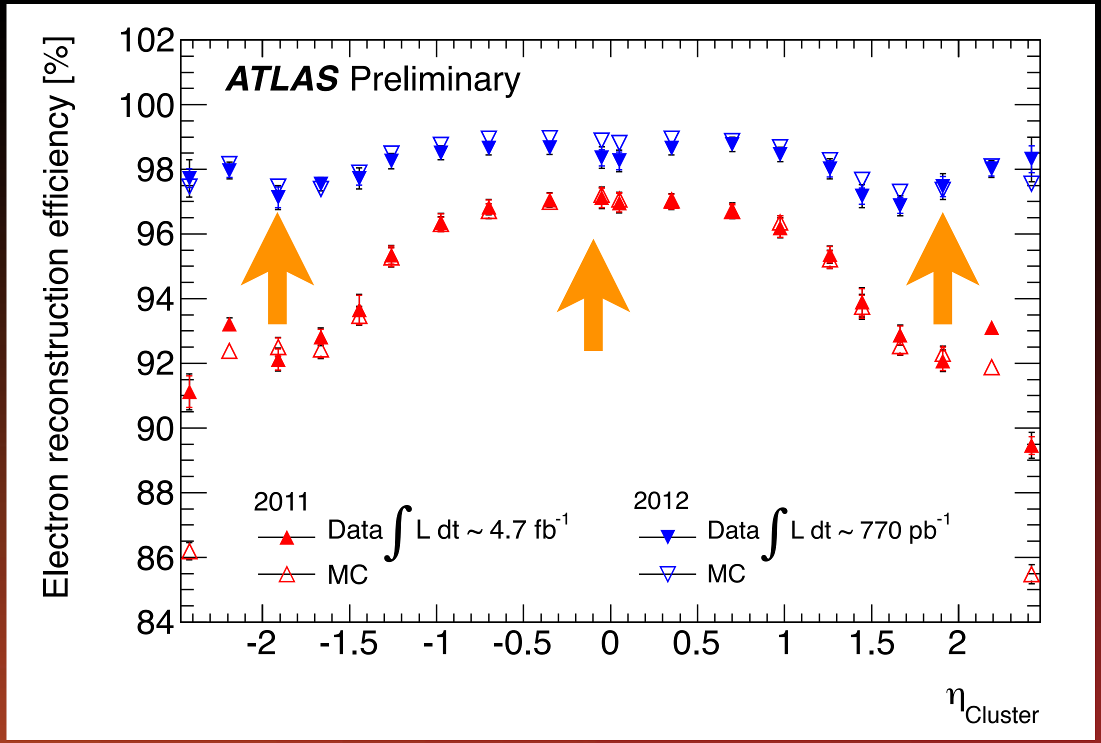




Tracking with Electron Brem. Recovery

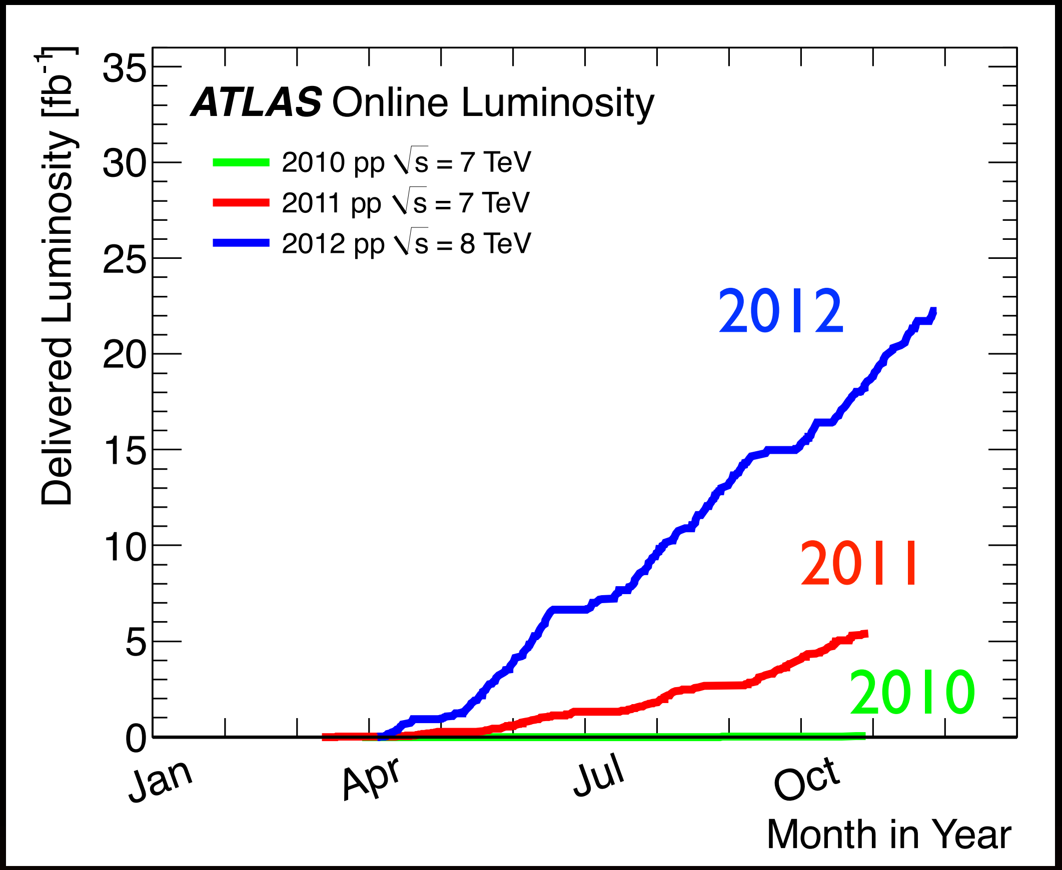
- bremsstrahlung in material
  - ➡ significant inefficiency in electron tracking
  - ➡ especially at low p_T (< 15 GeV)
    - limiting factor for $H \rightarrow ZZ^* \rightarrow 4e$
- strategy for brem. recovery
  - ➡ restrict recovery to regions pointing to electromagnetic clusters
  - ➡ pattern: allow for large energy loss in combinatorial Kalman filter
    - adjust noise term for electrons
  - ➡ global-χ^2 fitter allows for brem. point
  - ➡ adapt ambiguity processing (etc.) to ensure e.g. b-tagging is not affected
  - ➡ use full fledged Gaussian-Sum Filter in electron identification code
- most recent tracking update deployed in 2012
  - ➡ significant efficiency gain for Higgs discovery

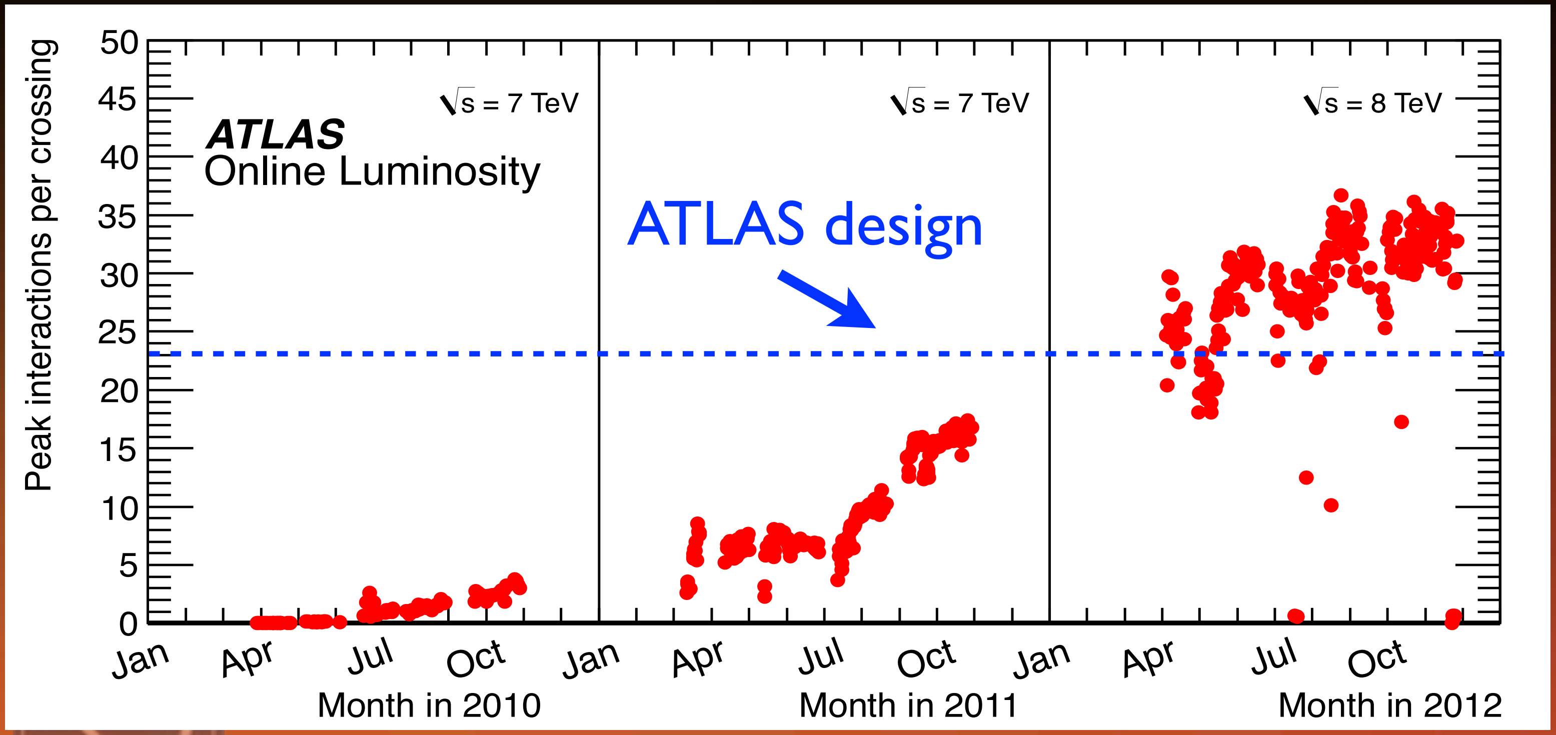


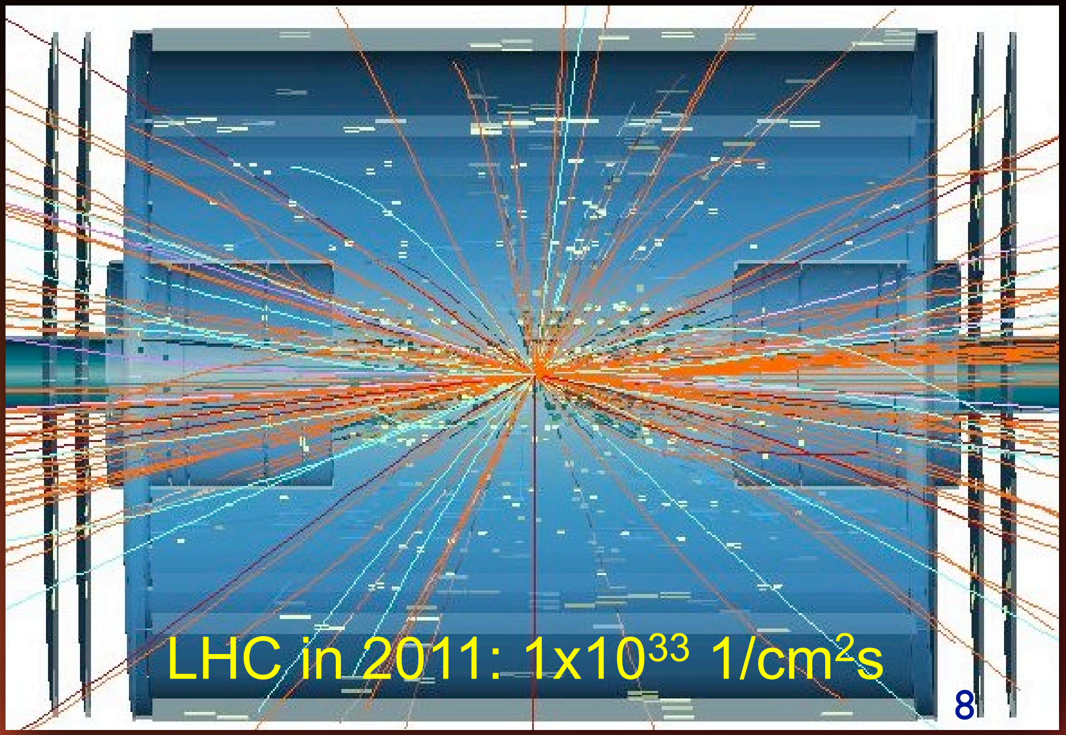


LHC is doing fantastically well

- 2012 operation
  ➡ peak event pileup routinely exceeding design values
- event pileup and other induced effects (e.g. radiation damage)
  ➡ challenge for the detector, T/DAQ and offline
    - so far ATLAS is doing very well

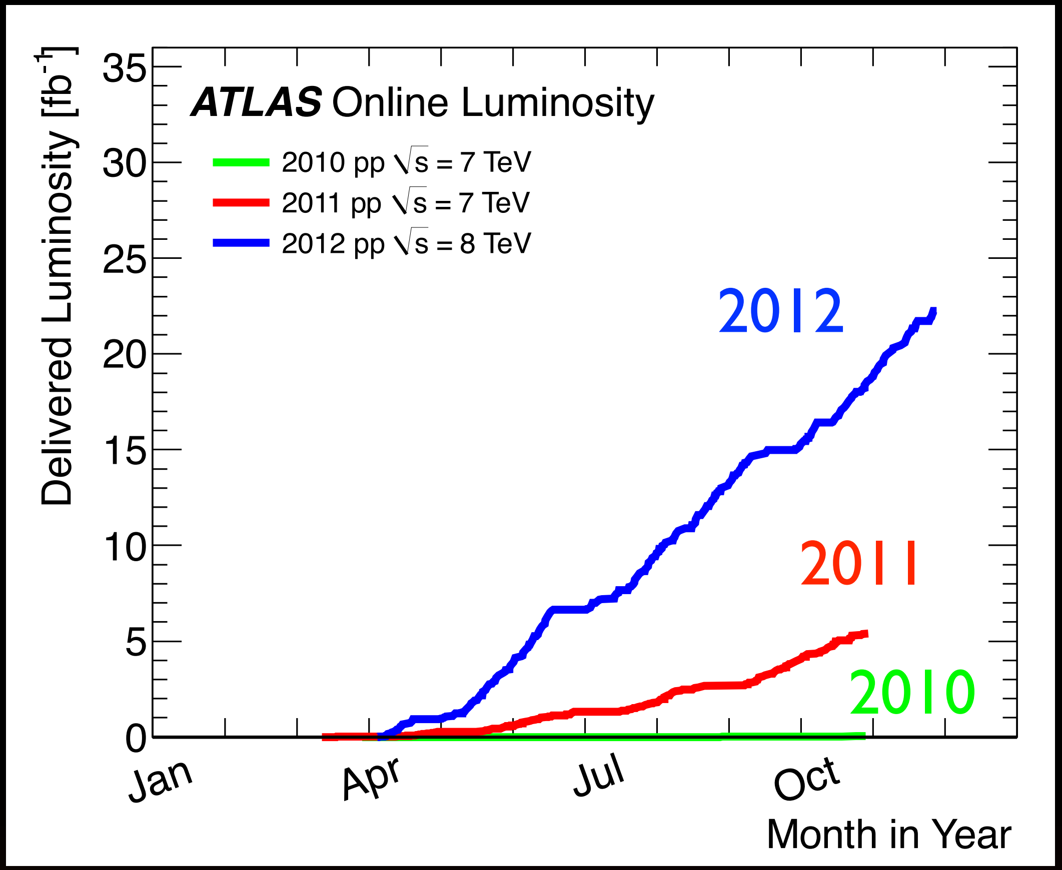


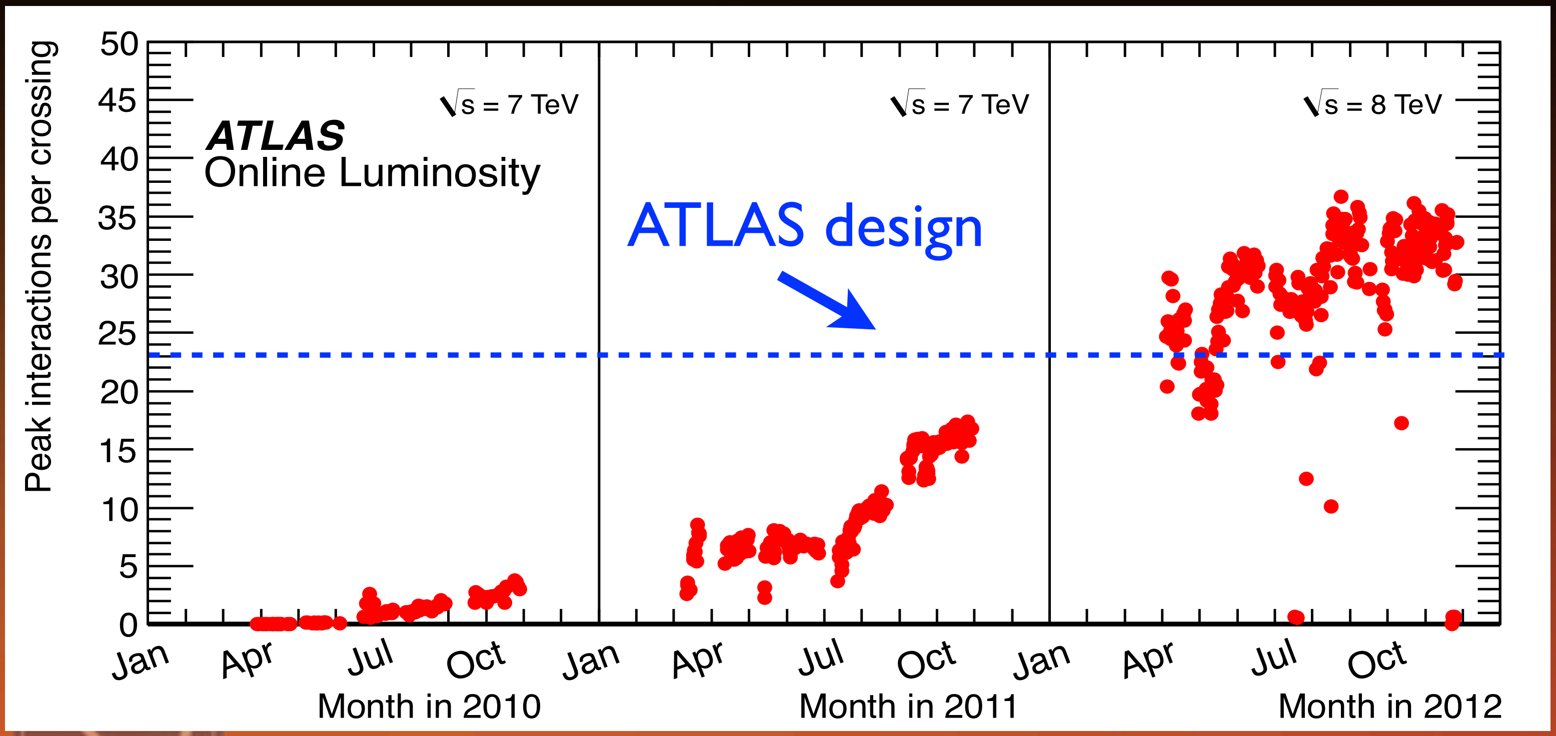


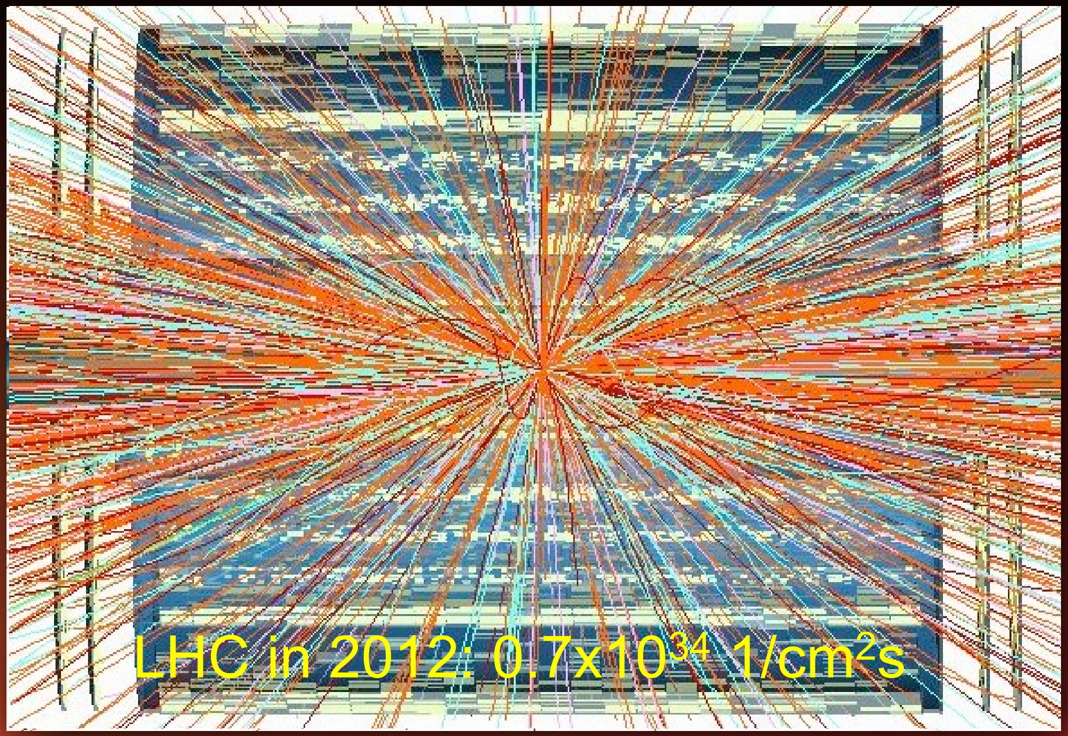


LHC is doing fantastically well

- 2012 operation
  - ➡ peak event pileup routinely exceeding design values
- event pileup and other induced effects (e.g. radiation damage)
  - ➡ challenge for the detector, T/DAQ and offline
    - so far ATLAS is doing very well



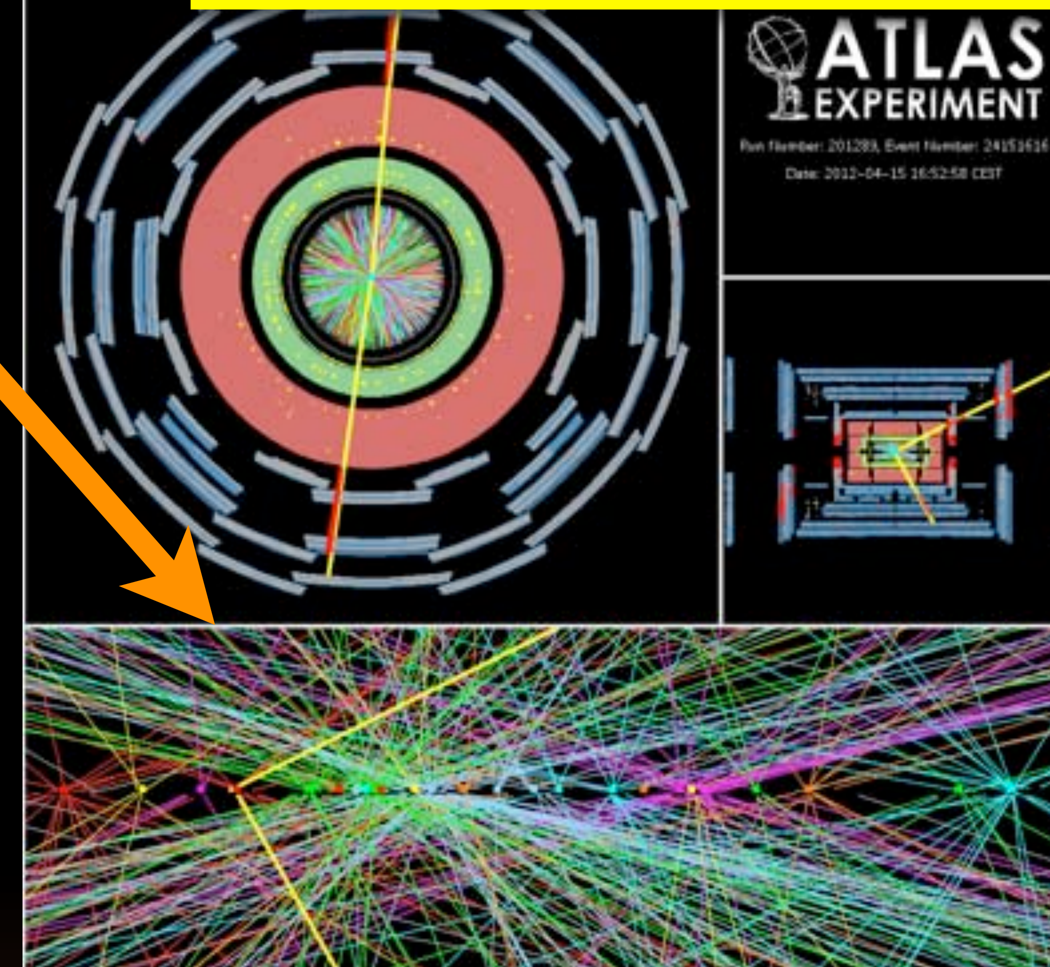




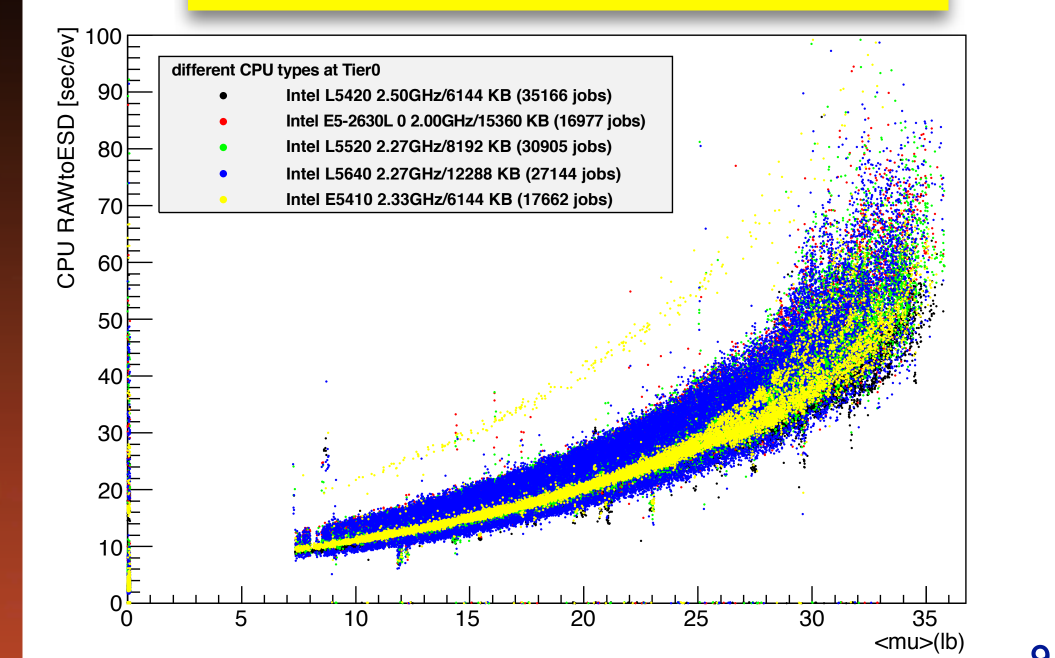
High Luminosity comes at a Price

- typical LHC event in 2012
  - ➡ high level of event pileup
- challenge for the experiments
  - ➡ trigger: select interesting interactions, keeping acceptable total rate
  - ➡ data volume: from the detector recorded on tape and to be processed/analyzed on computing GRID worldwide
  - ➡ reconstruction and analysis: make sense out of these very complex events and extracting interesting physics information
- huge development effort
  - ➡ already during shutdown 2011/2012
  - ➡ reconstruction resource driver: tracking !
- motivation for upgrade program:
  - ➡ preserve and improve physics and technical performance to fully benefit from increasing luminosity

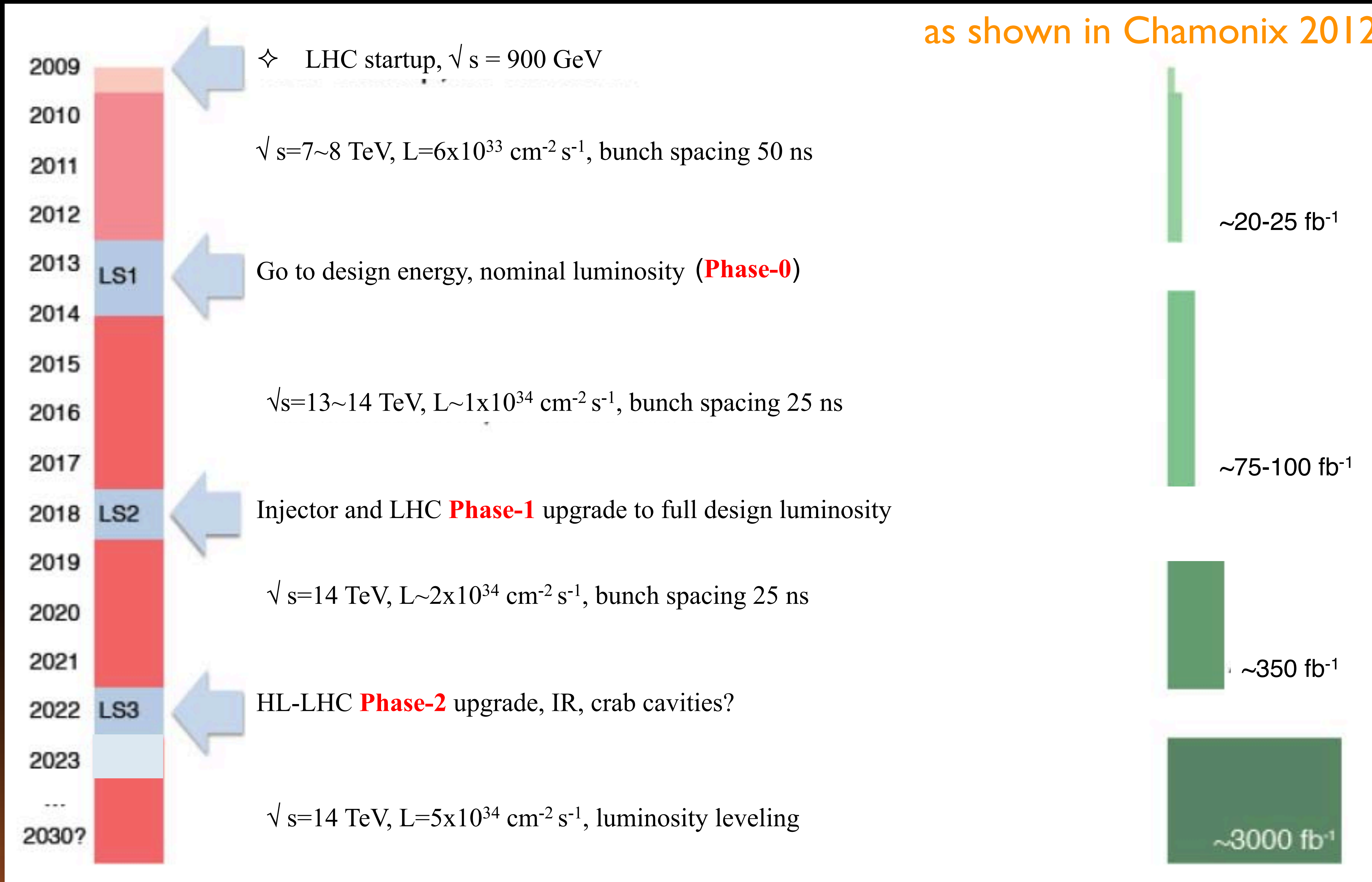
Z →μμ event with 25 reconstructed vertices



Tier-0 reconstruction time in 2012

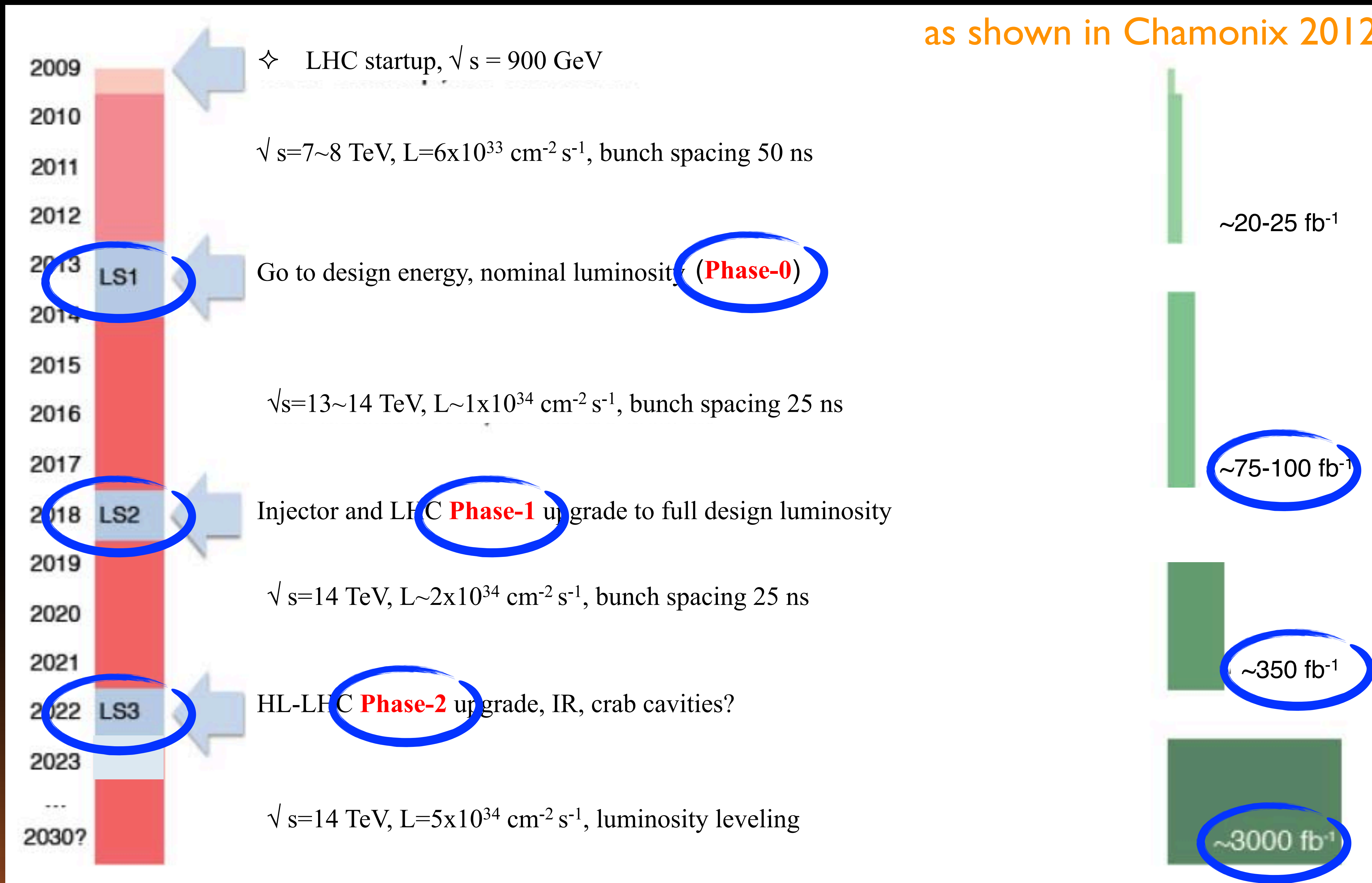


Upgrade Schedule Assumptions



➡ several tracking related updates planned

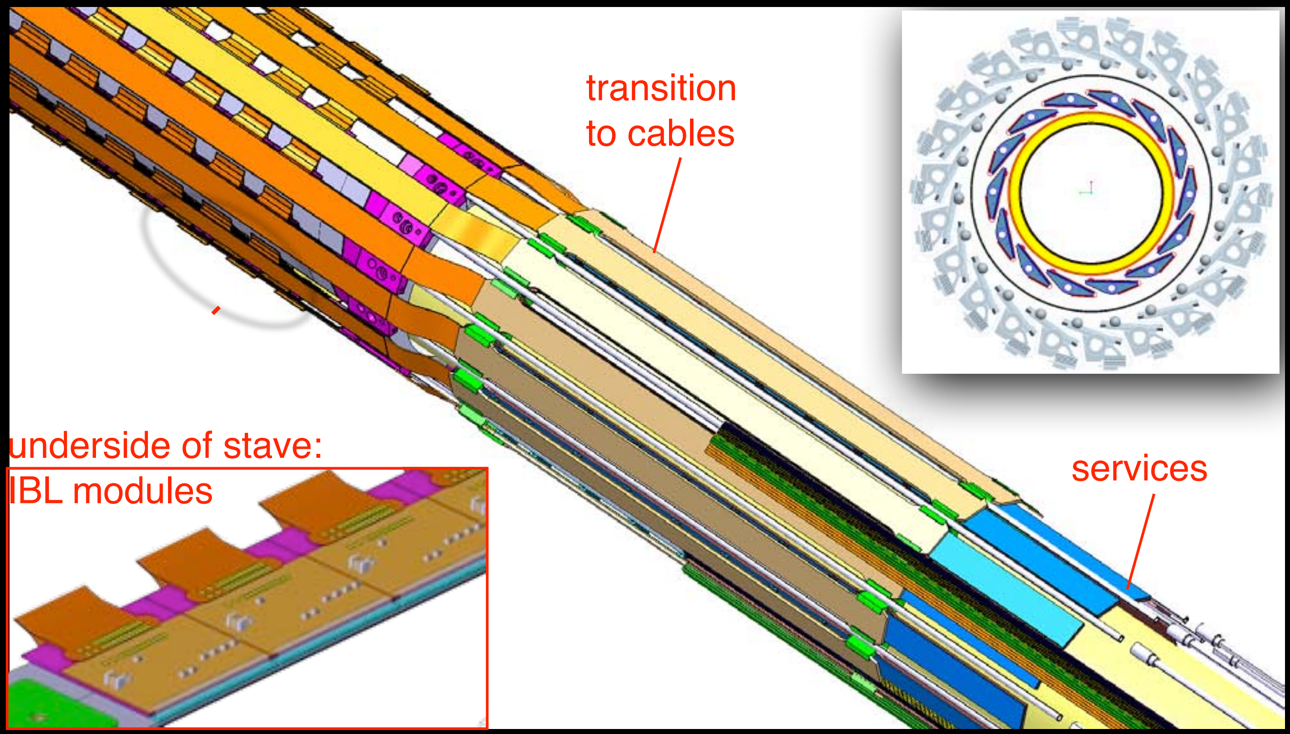
Upgrade Schedule Assumptions

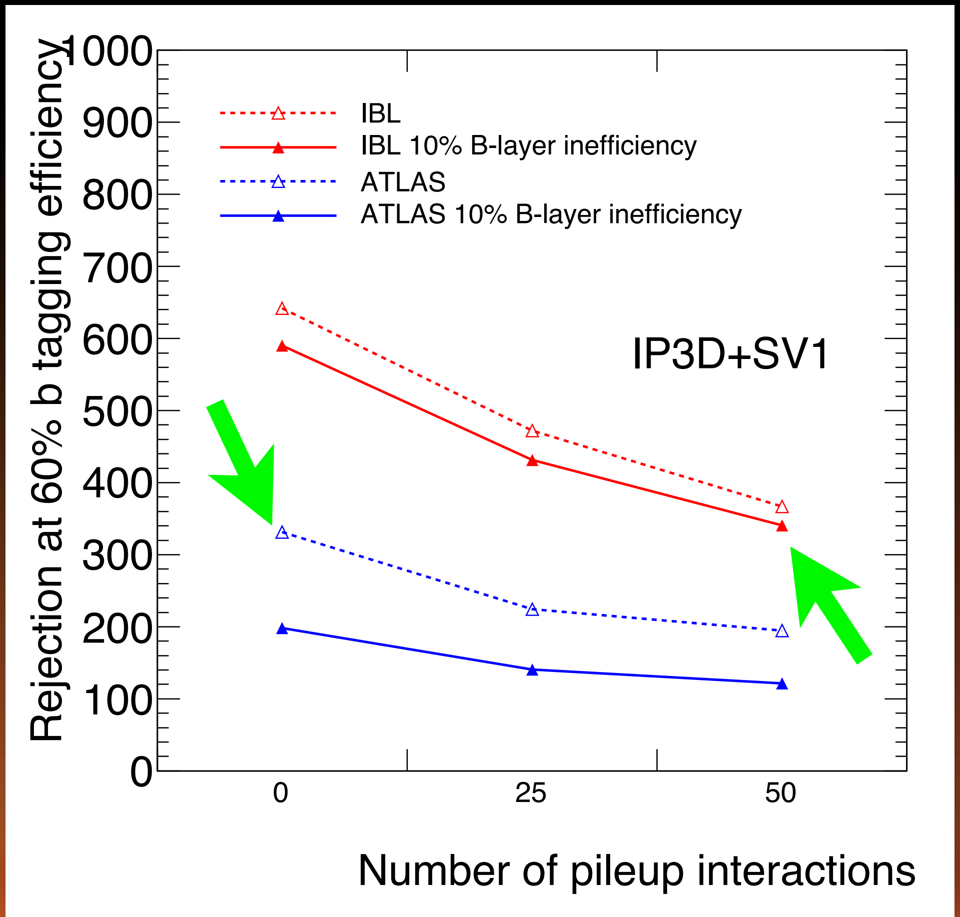


➡ several tracking related updates planned

Insertable B Layer (IBL)

- 4th pixel layer for Phase-0
  - ➡ add low mass layer closer to beam, with smaller pixel size
    - improve tracking, vertexing, b-tagging and τ-reconstruction
  - ➡ recovers from defects, especially in present b-layer
  - ➡ FE-I4b overcomes bandwidth limitations of present FE-I3



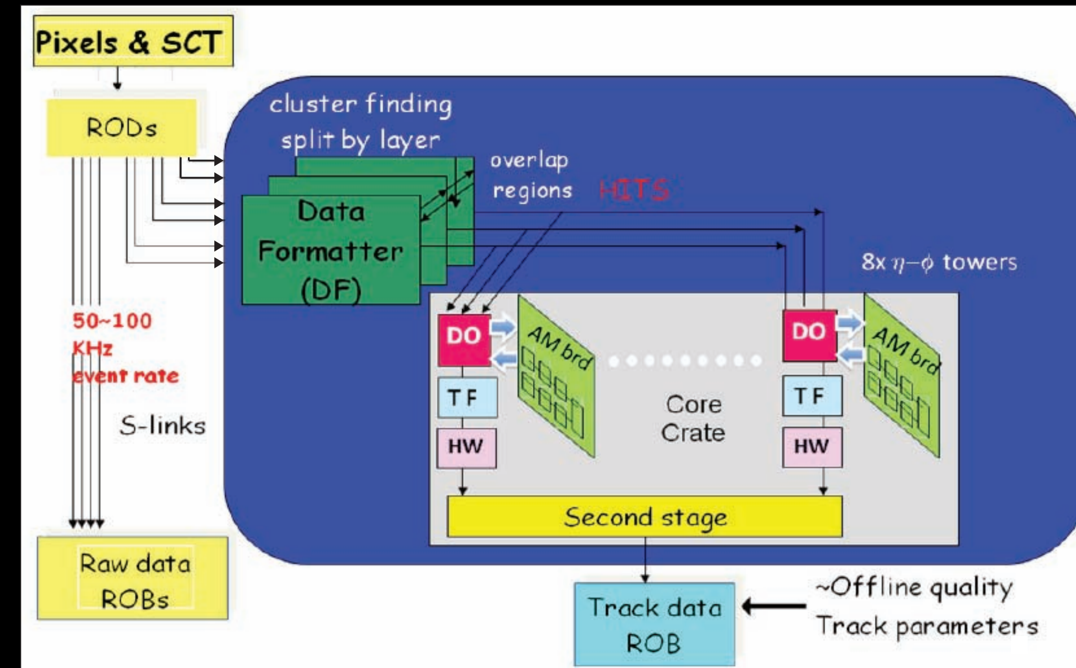


- IBL key specifications:
  - ➡ 14 staves, <R> = 33.25 *mm*
  - ➡ CO2 cooling, T < -15°C @ 0.2 *W/cm²*
  - ➡ X/X0 < 1.5 % (B-layer is 2.7 %)
  - ➡ 50 *µm* x 250 *µm* pixels (**planar** and **3D** sensors)
  - ➡ 1.8° overlap in ϕ, < 2% gaps in Z
  - ➡ 32/16 single/double FE-I4 modules per stave
  - ➡ radiation tolerance $5 \cdot 10^{15}$ *neq/cm²*
- mounted on new beam pipe
  - ➡ installation options still to be decided
  - ➡ may extract present Pixel Detector to replace nSQPs (decision this year)

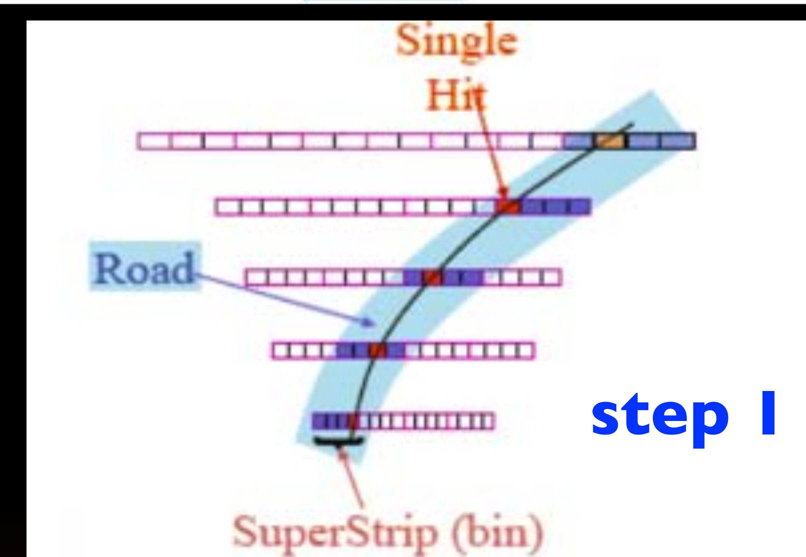
stave -1 after loading

The Fast Tracker (FTK)

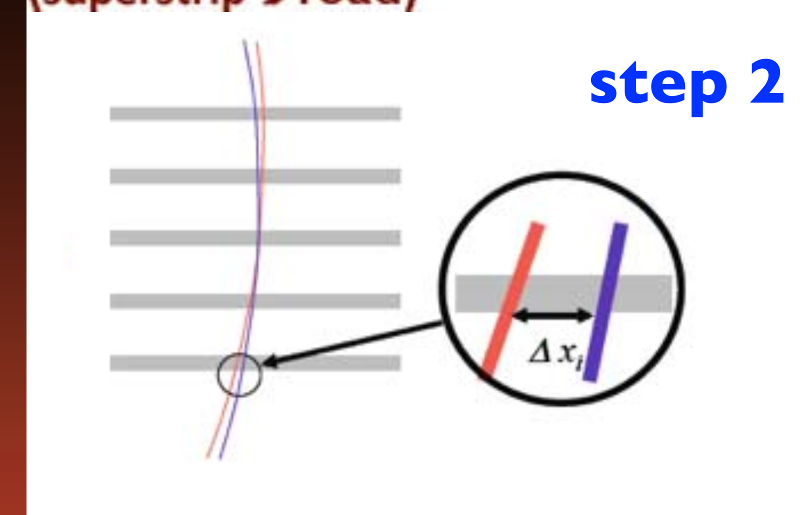
- current ATLAS trigger chain
  - ➡ Level-1: hardware based (~50 *kHz*)
  - ➡ Level-2: software based with RoI access to full granularity data (~5 *kHz*) ← tracking enters here
  - ➡ Event Filter: software trigger (~500 *Hz*)



- FTK: hardware based tracking for Phase-1
  - ➡ descendent of the CDF Silicon Vertex Trigger (SVT)
  - ➡ inputs from Pixel and SCT
    - data in parallel to normal read-out
  - ➡ two step reconstruction
    - associative memories for parallel pattern finding
    - linearized track fit implemented in FPGAs
  - ➡ provides track information to Level-2 in ~ 25 *μs*



Pattern recognition in coarse resolution (superstrip➔road)

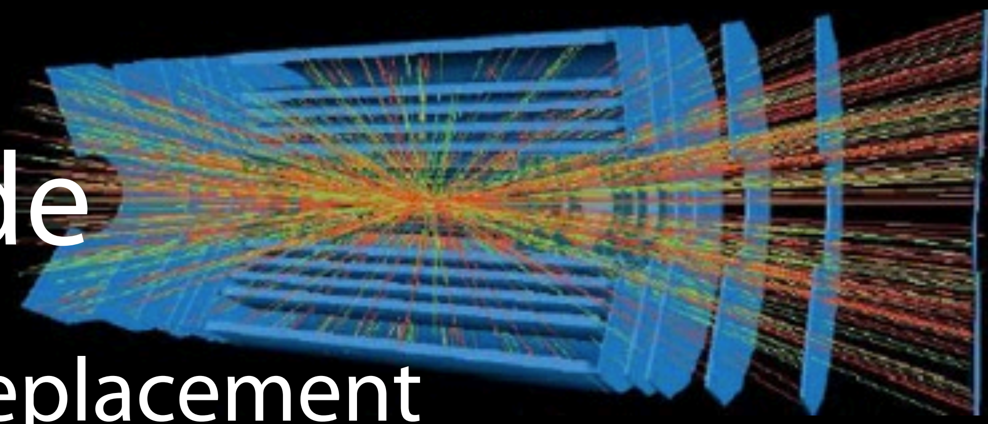


Track fit in full resolution (hits in a road)

$F(x_1, x_2, x_3, \ldots) \sim a_0 + a_1\Delta x_1 + a_2\Delta x_2 + a_3\Delta x_3 + \ldots = 0$

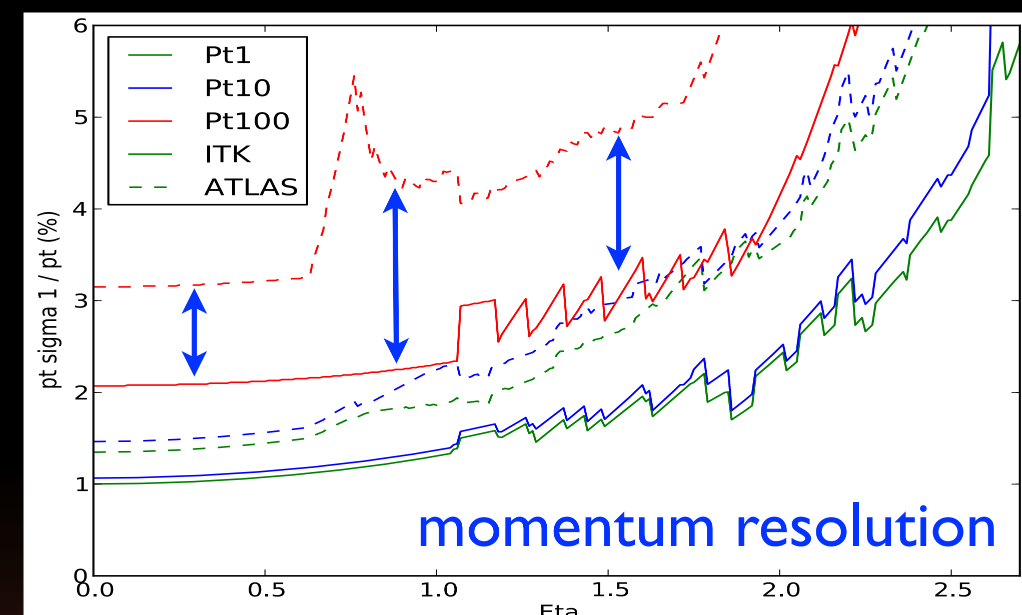
- FTK: trigger goals
  - ➡ lepton isolation, b-tagging, τ-reconstruction
  - ➡ primary vertex reconstruction, vertex counting
  - ➡ pileup robustness of (track based) MET and jet triggers

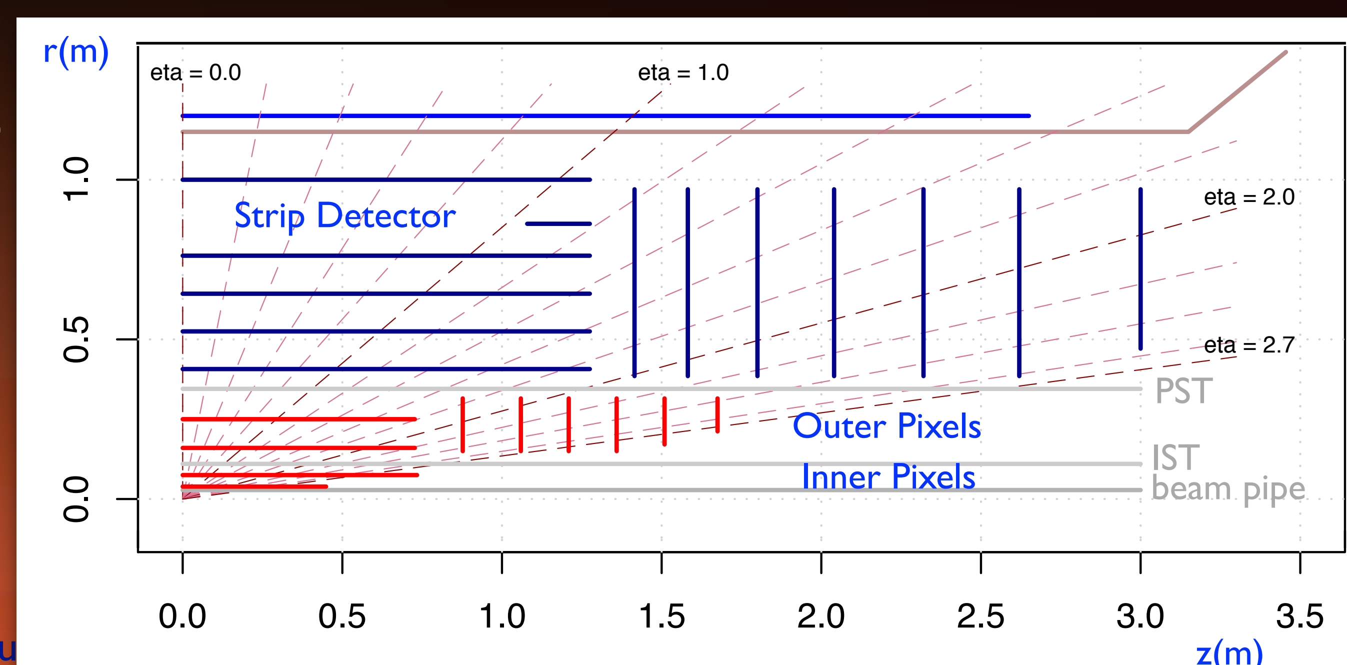
Phase-2 Inner Tracker Upgrade



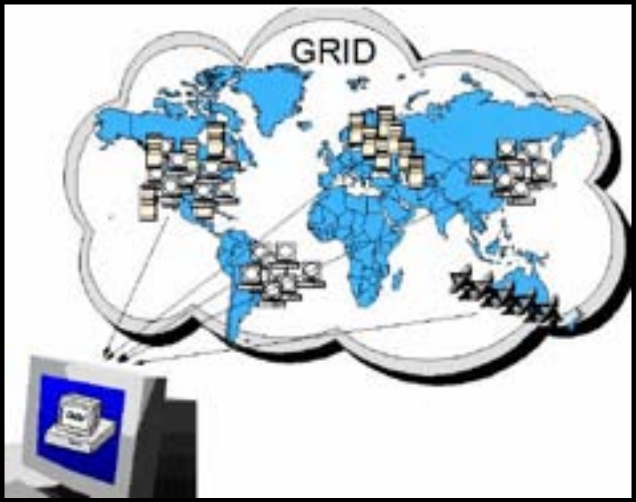
- to keep ATLAS running requires tracker replacement
  - ➡ current tracker designed to survive up to 10 *MRad* in strip detectors (≤ 700 fb^{-1})
  - ➡ replace with an all silicon tracker to match the challenge of 140-200 pileup events

- main ITK design parameters
  - ➡ **Inner Pixels:**
    - 2 replaceable layers close to enlarged Phase-2 beam pipe
    - smaller pixel pitch to improve b-tagging (FE-I5)
  - ➡ **Outer Pixels:**
    - 2 barrel layers at increased radii to improve tracking in jets
    - pixel endcaps ensure full tracking coverage to η=2.5
    - some standalone tracking capability to η=2.7 (muons)
  - ➡ **Strip Detector:**
    - maximize momentum resolution (*B·dl*)
    - double sided strips in 5 layer, 7 disk, plus stub
    - shorter strips close to PST to limit occupancy
  - ➡ overall a 14 hit system down to η=2.5
    - robustness, avoid fakes at high pileup
    - overall much reduced material budget
      - ➡ plan is to add Level-1 track trigger
        - in a Level-0/Level-1 scheme
        - FTK like hardware tracking



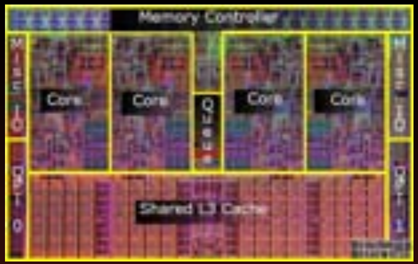


Computing and Offline

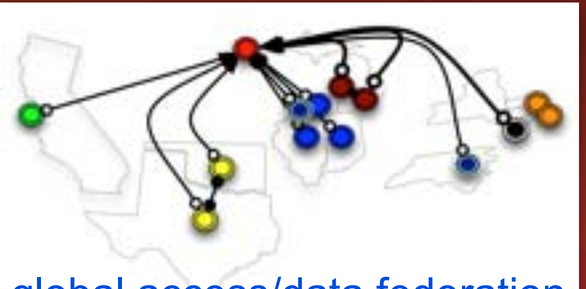


- vital part of the upgrade program
  - ➡ support upgrade with detector simulation
  - ➡ upgrade of the computing and offline software infrastructure

- many challenges ahead
  - ➡ computing infrastructure is constantly evolving
    - GRID middleware, cloud computing, storage systems, networking...
  - ➡ increasing integrated luminosity, trigger rates and event sizes
    - ATLAS Production System and Data Management needs to scale
    - GRID luminosity for simulation is becoming rapidly a factor
  - ➡ reconstruction needs to cope with even higher levels of event pileup

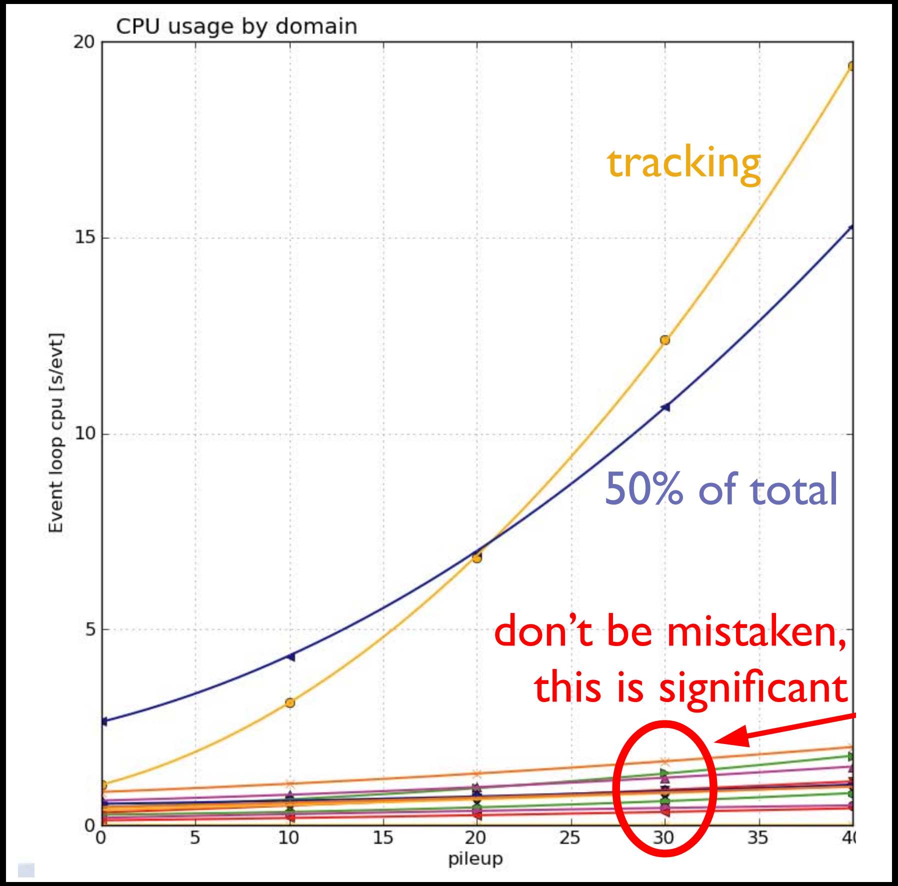


- upgrade on the fly, while experiment is operating
- industry may move to new technologies
  - ➡ many-core architectures may replace present X86 boxes (*a la* Intel MIC)
  - ➡ need to be prepared to adapt or re-implement large parts of framework as well as offline (and high level trigger) software chain
- formally part of Phase-2 Letter of Intent
  - ➡ but LS1 shutdown is unique window of opportunity


global access/data federation

CPU Performance vs Pileup

- tracking is driving CPU requirements
  - ➡ scaling with pileup is fastest on average
- LS1 preparation for Phase-0
  - ➡ need to significantly gain in CPU
    - tension: physics vs technical performance and tendency to use more fancy tools (e.g. GSF)
  - ➡ only last resort is cutting harder on tracks

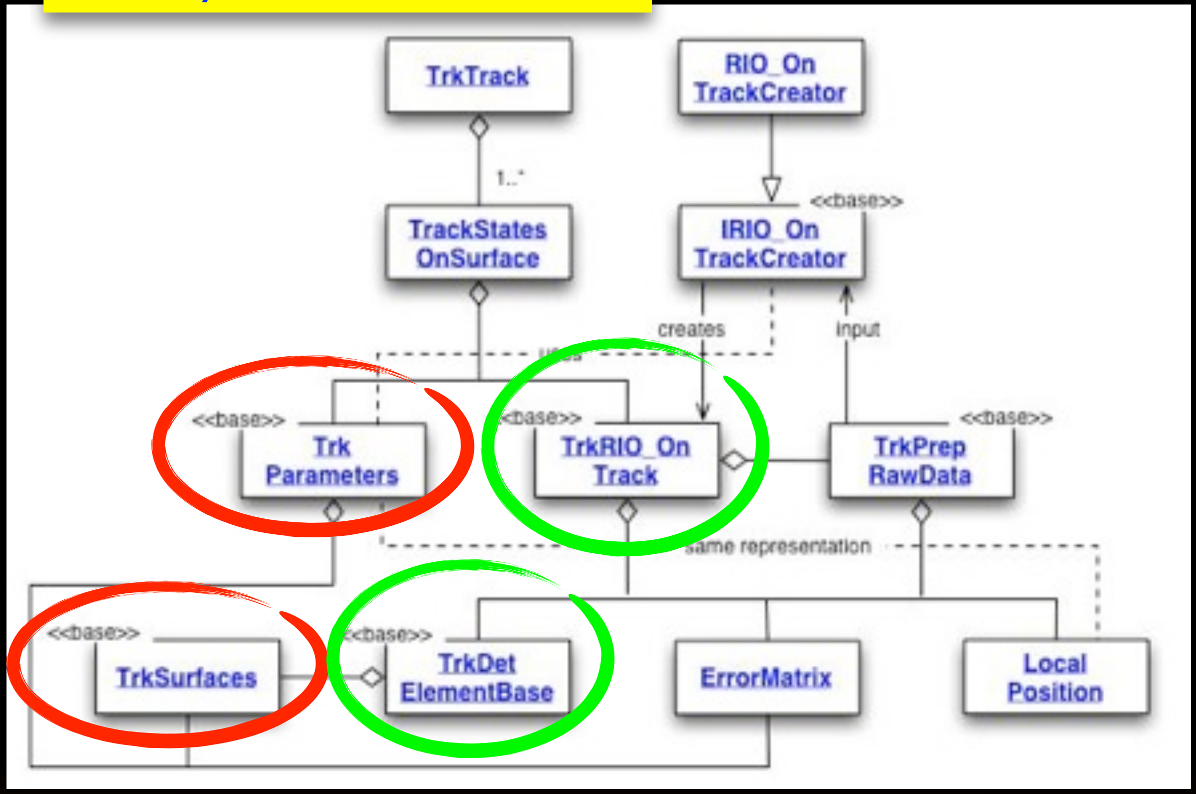


- started development program
  - ➡ review event data model (EDM) and use of malloc
    - present EDM and algorithm design is very OO centric, causing overheads
    - EDM objects are often scattered in memory, causing cache faults
  - ➡ explore (auto-)vectorization and multithreading
    - vectorization: expect factors > 2 for mathematical algorithms
    - multithreading: allows to use more cores with less total memory
    - but: precision tracking is a lot about decide and branch...
  - ➡ another iteration in algorithmic optimization
    - try to identify and replace inefficient algorithmic code (but its already optimized)

ATLAS Tracking Event Data Model

- current EDM dates back ~ 8 years
  - ➡ very much influenced by OO design ideas
    - strong typing, heavily polymorph
  - ➡ needed to support all existing applications
    - Inner Detector, Muon Spectrometer, Trigger
  - ➡ functionality added since
    - especially persistency with schema support

EDM system of base-classes:



- most CPU demanding algorithms
  - ➡ internally use data pools and simplified EDM

- EDM redesign for LS1
  - ➡ remove EDM layers to support unused reconstruction functionality
  - ➡ extrapolation engine migrates fully to curvilinear representation
  - ➡ deduce inheritance and optimize memory layout
    - enable (more) general use of data pools
    - arrange private data to better support e.g. GPUs or vectorization (?)
  - ➡ replace CLHEP with vector, geometry and math library that fully supports vectorization (needs R&D)

Vectorizing Tracking SW

- algorithmic tracking code
  - ➡ lots of vector algebra, trigonometric functions, floating point operations, ...
  - ➡ natural candidates for (auto-)vectorization
- ATLAS has complex B-field
  - ➡ field transport is hot-spot
    - in simulation and reconstruction
  - ➡ developed state of the art modified Runge-Kutta-Nystrom techniques
    - some variants are part of recent G4 releases

loop in Runge-Kutta::Step:

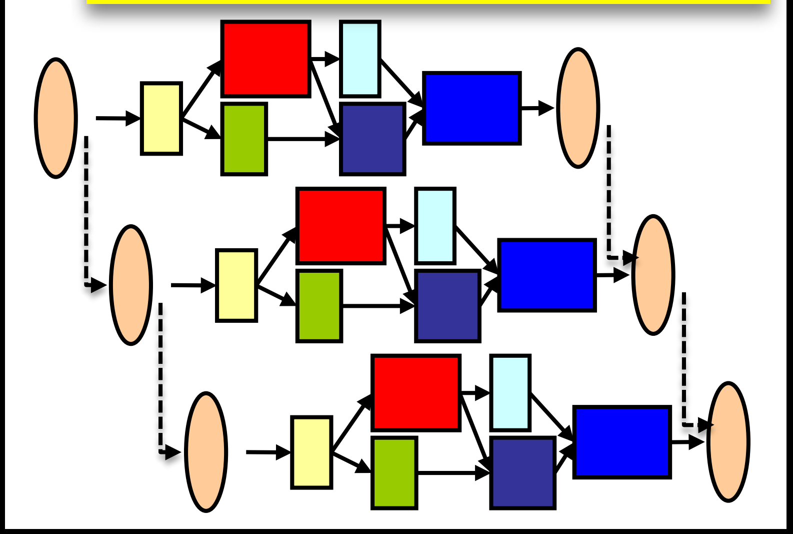
```
for(int i=0; i<42; i+=7) {
    double* dR   = &P[i];
    double* dA   = &P[i+3];

    double dA0   = H0[ 2]*dA[1]-H0[ 1]*dA[2];
    double dB0   = H0[ 0]*dA[2]-H0[ 2]*dA[0];
    double dC0   = H0[ 1]*dA[0]-H0[ 0]*dA[1];

    if(i==35) {dA0+=A0; dB0+=B0; dC0+=C0;}

    double dA2   = dA0+dA[0];
    double dB2   = dB0+dA[1];
    double dC2   = dC0+dA[2];

    double dA3   = dA[0]+dB2*H1[2]-dC2*H1[1];
    double dB3   = dA[1]+dC2*H1[0]-dA2*H1[2];
    double dC3   = dA[2]+dA2*H1[1]-dB2*H1[0];

    if(i==35) {dA3+=A3-A00; dB3+=B3-A11; dC3+=C3-A22;}

    double dA4   = dA[0]+dB3*H1[2]-dC3*H1[1];
    double dB4   = dA[1]+dC3*H1[0]-dA3*H1[2];
    double dC4   = dA[2]+dA3*H1[1]-dB3*H1[0];

    if(i==35) {dA4+=A4-A00; dB4+=B4-A11; dC4+=C4-A22;}

    double dA5   = dA4+dA4-dA[0];
    double dB5   = dB4+dB4-dA[1];
    double dC5   = dC4+dC4-dA[2];

    double dA6   = dB5*H2[2]-dC5*H2[1];
    double dB6   = dC5*H2[0]-dA5*H2[2];
    double dC6   = dA5*H2[1]-dB5*H2[0];

    if(i==35) {dA6+=A6; dB6+=B6; dC6+=C6;}

    dR[0]+=(dA2+dA3+dA4)*S3; dA[0]=(dA0+dA3+dA3+dA5+dA6)*.33333333;
    dR[1]+=(dB2+dB3+dB4)*S3; dA[1]=(dB0+dB3+dB3+dB5+dB6)*.33333333;
    dR[2]+=(dC2+dC3+dC4)*S3; dA[2]=(dC0+dC3+dC3+dC5+dC6)*.33333333;
}
```

- gcc 4.7 failed to auto-vectorize Runge-Kutta::Step
  - ➡ manual vectorization gave speedup by factor 2.4 (SSE) on Sandy Bridge
- underlines importance of new math/vector library
  - ➡ will ease (auto-)vectorization of code

Multithreading

- make use of many core architectures
  - ➡ reduce required memory per core
  - ➡ future algorithm level concurrency support (Gaudi?)

Gaudi concurrency proposal



- R&D for parallel (GPU) tracking algorithms
  - ➡ aim is GPU replacement of CPU intensive algorithms
    - usually significant approximations are required
  - ➡ Level-2 tracking most complex prototype so far
    - better suited for this approach, see later...

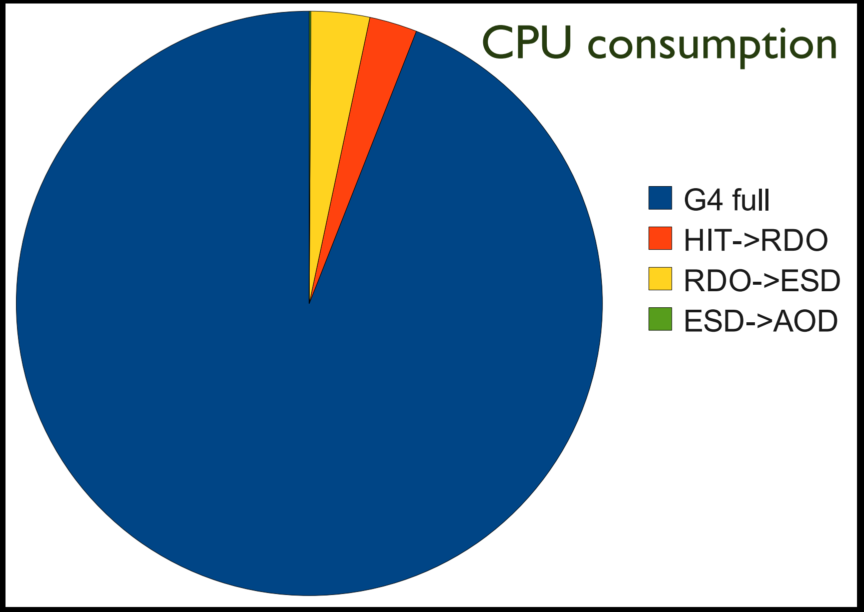
- full fledged offline tracking ?
  - ➡ so no shortcuts
    - much more difficult problem
    - especially, same physics performance !
  - ➡ first prototype to run full tracking chain
    - using POSIX or TBB, in future will have framework support
  - ➡ first experimental results encouraging, but a long way still to go

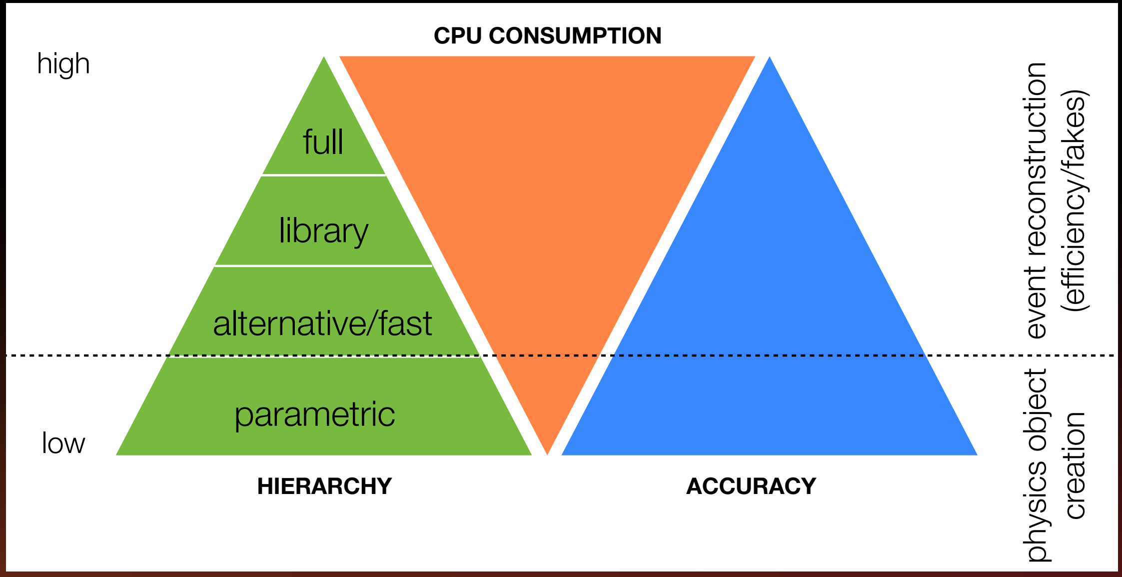
full offline combinatorial track finder

Number threads vs time in sec	0 pileup	10 pileup	20 pileup	30 pileup	40 pileup
1	.0873	.4717	1.086	2.037	3.837
2	.0503 (1.74)	.2611 (1.81)	.5883 (1.85)	1.092 (1.87)	2.069 (1.85)
3	.0407 (2.14)	.1898 (2.48)	.4341 (2.50)	.7928 (2.57)	1.476 (2.60)
4	.0349 (2.50)	.1626 (2.90)	.3546 (3.05)	.6688 (3.05)	1.265 (3.03)

A few Words on Simulation

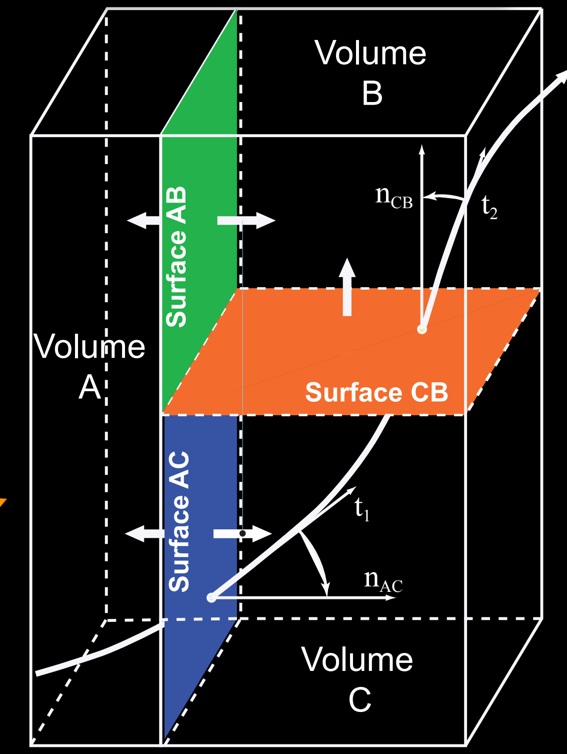
- GRID Monte Carlo "luminosity"
  - ➡ limited by CPU needs for G4 in ATLAS
- full fledged G4 based simulation
  - ➡ yields best description of detector response
  - ➡ GRID "luminosity" will not scale with MC needs
- alternative simulation techniques
  - ➡ huge potential CPU gains, less accuracy
  - ➡ frozen shower libraries:
    - give large gains, still relatively detailed
  - ➡ parametric detector simulation:
    - usually not precise enough for physics
  - ➡ alternative methods of fast simulation:
    - fast calorimeter simulation
    - fast track simulation based on track reconstruction software framework





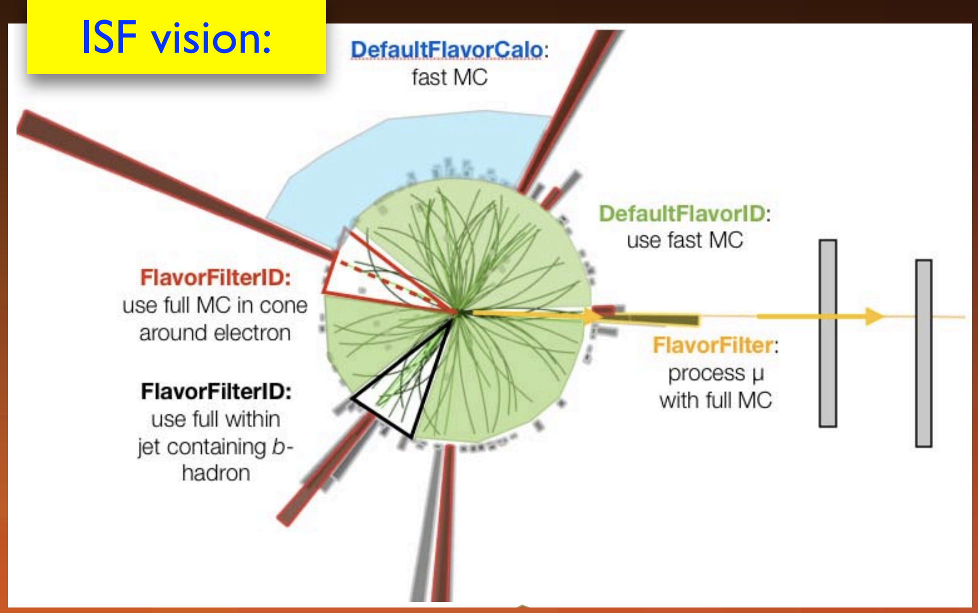
Fast Track Simulation and the ISF

embedded navigation:

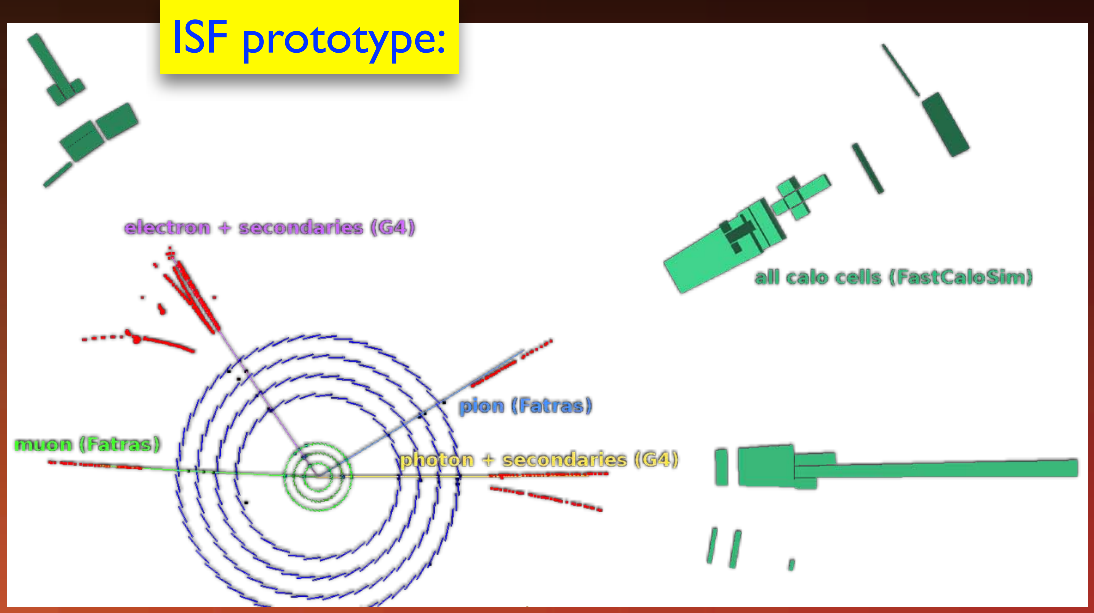


- track reconstruction framework
  - ➡ contains a transport engine, b-field and material geometry
  - ➡ naturally basis for fast simulation engine:
    - add particle stack and (fast) physics processes
  - ➡ benefit from fast track reconstruction techniques (e.g. navigation)
- ATLAS Integrated Simulation Framework (ISF)
  - ➡ within one event, choose simulation engines for different event aspects
    - i.e. use full simulation e.g. for a high-p_T b-jet and fast for underlying event
  - ➡ in fastest version digitization and reconstruction becomes bottleneck
    - extend scheme to cover full chain (fast digi. and fast reco. in regions)
    - possibly huge gains in overall CPU needs !

ISF vision:

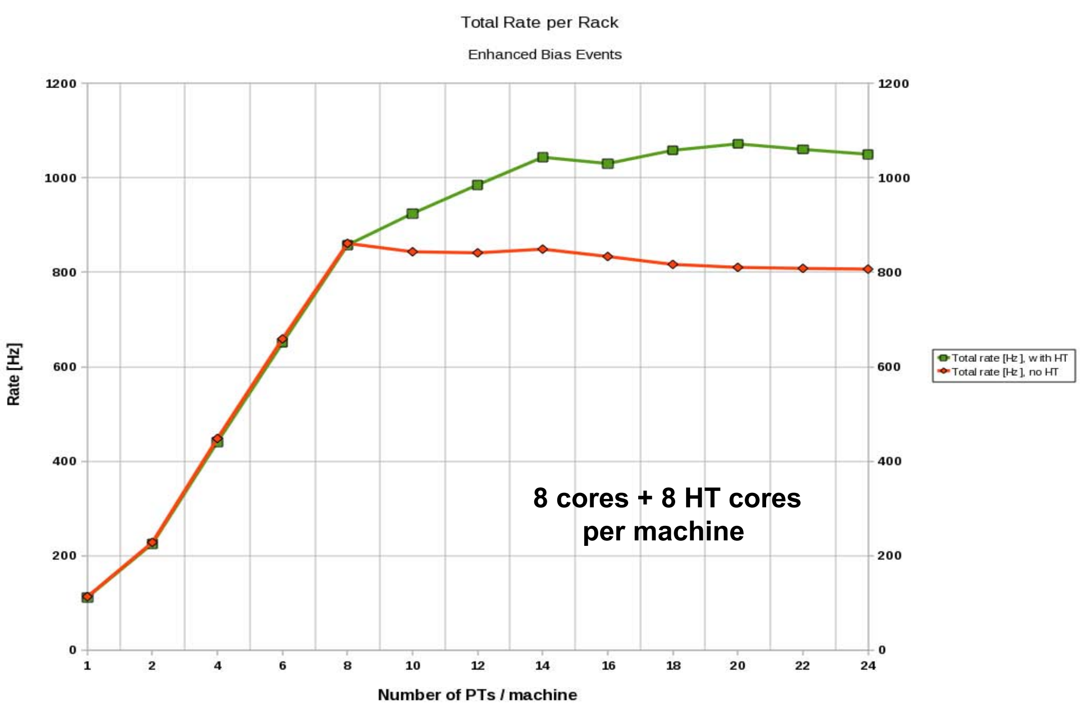


ISF prototype:



Trigger Upgrade and Tracking

- HLT algorithms share same code base as offline
  - ➡ will benefit automatically from offline developments and code optimization, including vectorization and support for multithreading
- Level-2 and Event Filter processing
  - ➡ event parallelism with multiple selection processes
    - currently does not require framework and offline code to be thread safe
    - like for offline, processor technology will require algorithms to go fully multithreaded
  - ➡ T/DAQ controlled applications like e.g. data flow are already heavily multithreaded

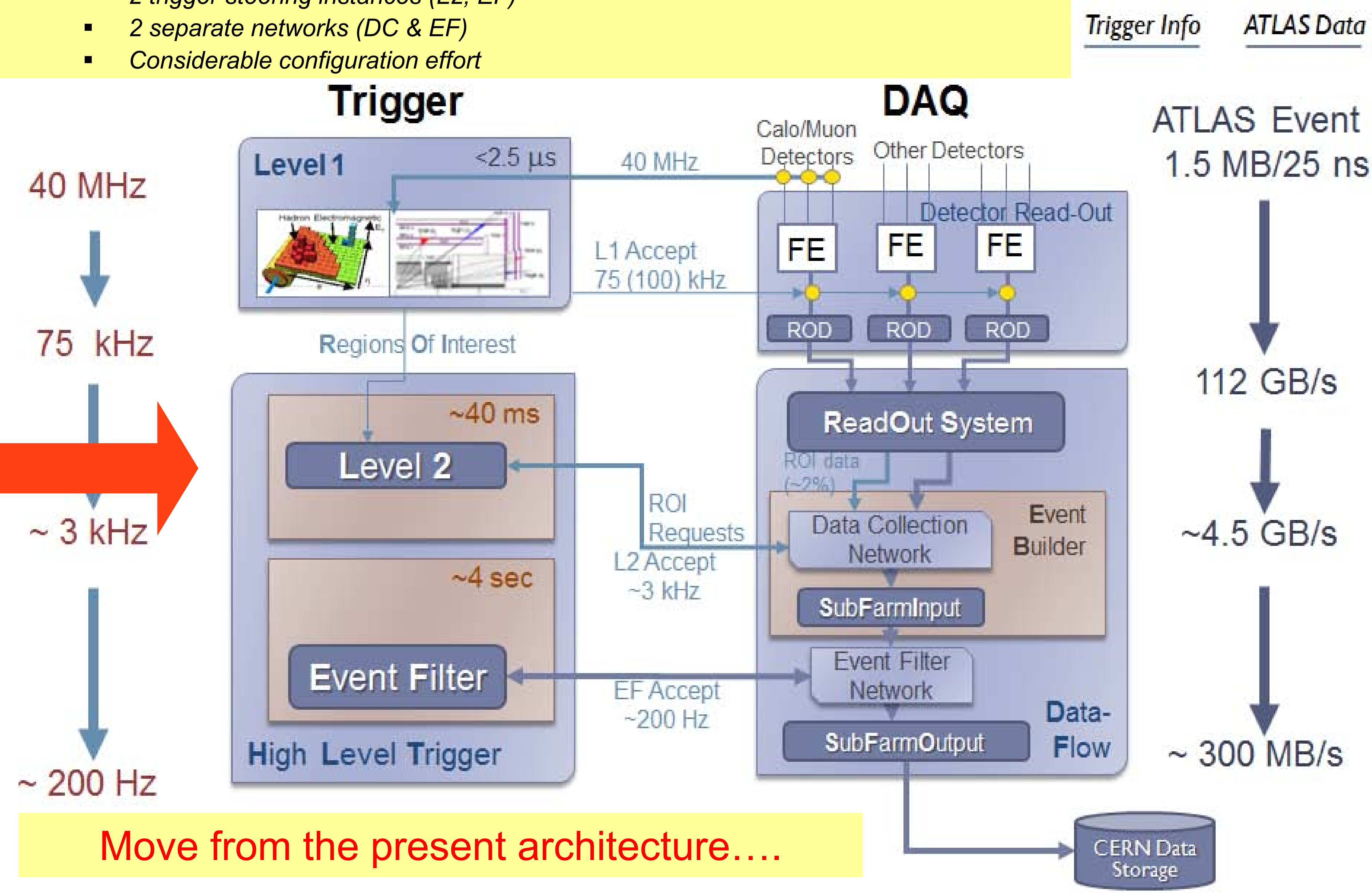


- evolution of the T/DAQ data flow architecture
  - ➡ will as well require better HLT and offline software integration
  - ➡ see next slides

Data Flow Evolution

Present architecture has many farm and network domains

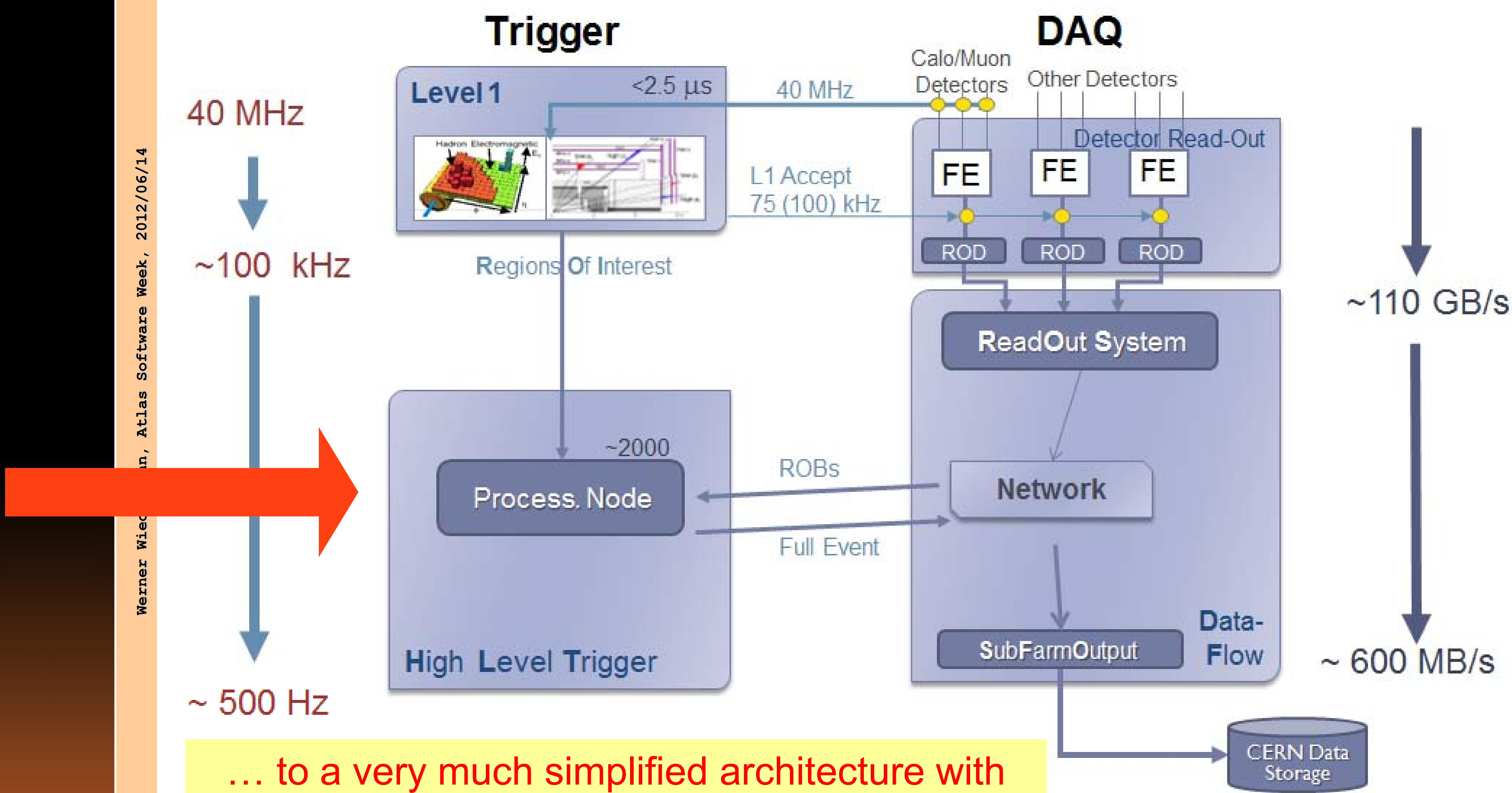
- *CPU and network resources have to be balanced for three different farms: L2, EB, EF*
- *2 trigger steering instances (L2, EF)*
- *2 separate networks (DC & EF)*
- *Considerable configuration effort*



Move from the present architecture….

Werner Wie…n, Atlas Software Week, 2012/06/14

Data Flow Evolution



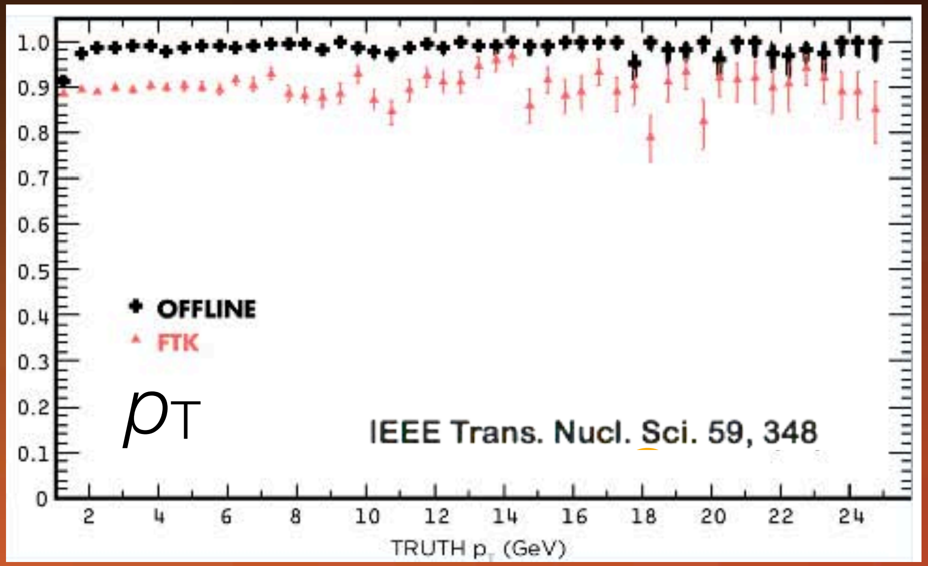
… to a very much simplified architecture with L2-, EB- and EF- functionality merged for each processing node

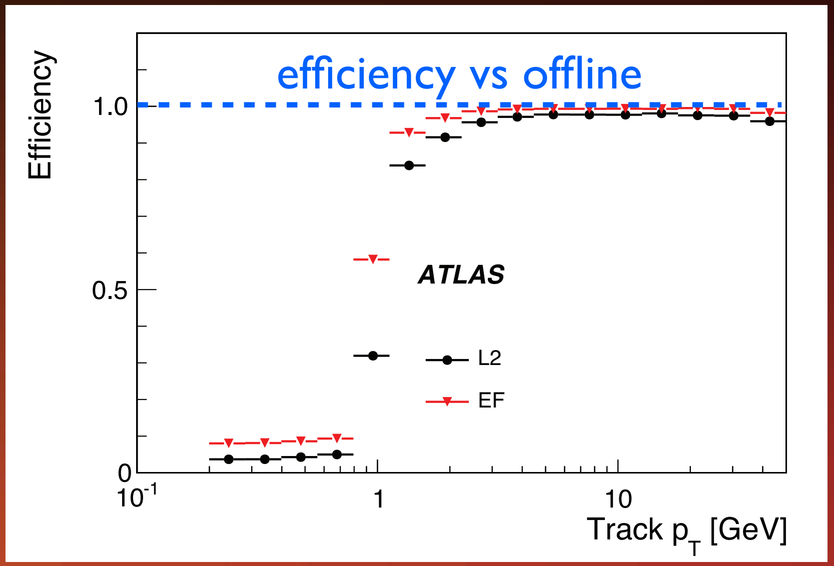
HLT and Offline Tracking Integration

- currently Level-2 and Event Filter run independently
  - ➡ 40 msec Level-2 latency compared to 2 sec for Event Filter require dedicated algorithms
  - ➡ new data flow: use Level-2 to seed Event Filter tracking in same process

FTK ??? → Level-2 → Event Filter → ???

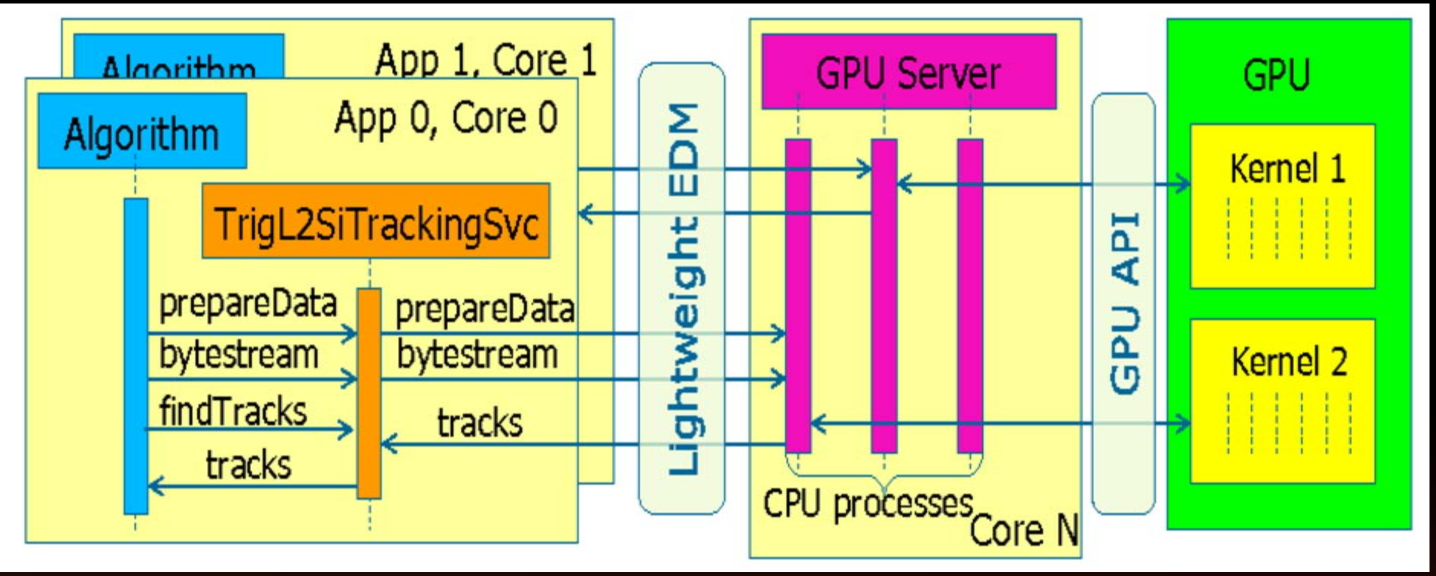
  - save CPU by reusing already decoded data, no need to redo seeding, but can use full fledged Event Filter algorithms to boost precision
  - could even use FTK tracks with cluster information as input to Level-2 fitter to replace Level-2 track seeding and candidate finding
- ➡ FTK/Level-2 tracking is compromise of efficiency vs technical performance
  - need to preserve e.g. Event Filter performance for b-tagging and τ-tracking





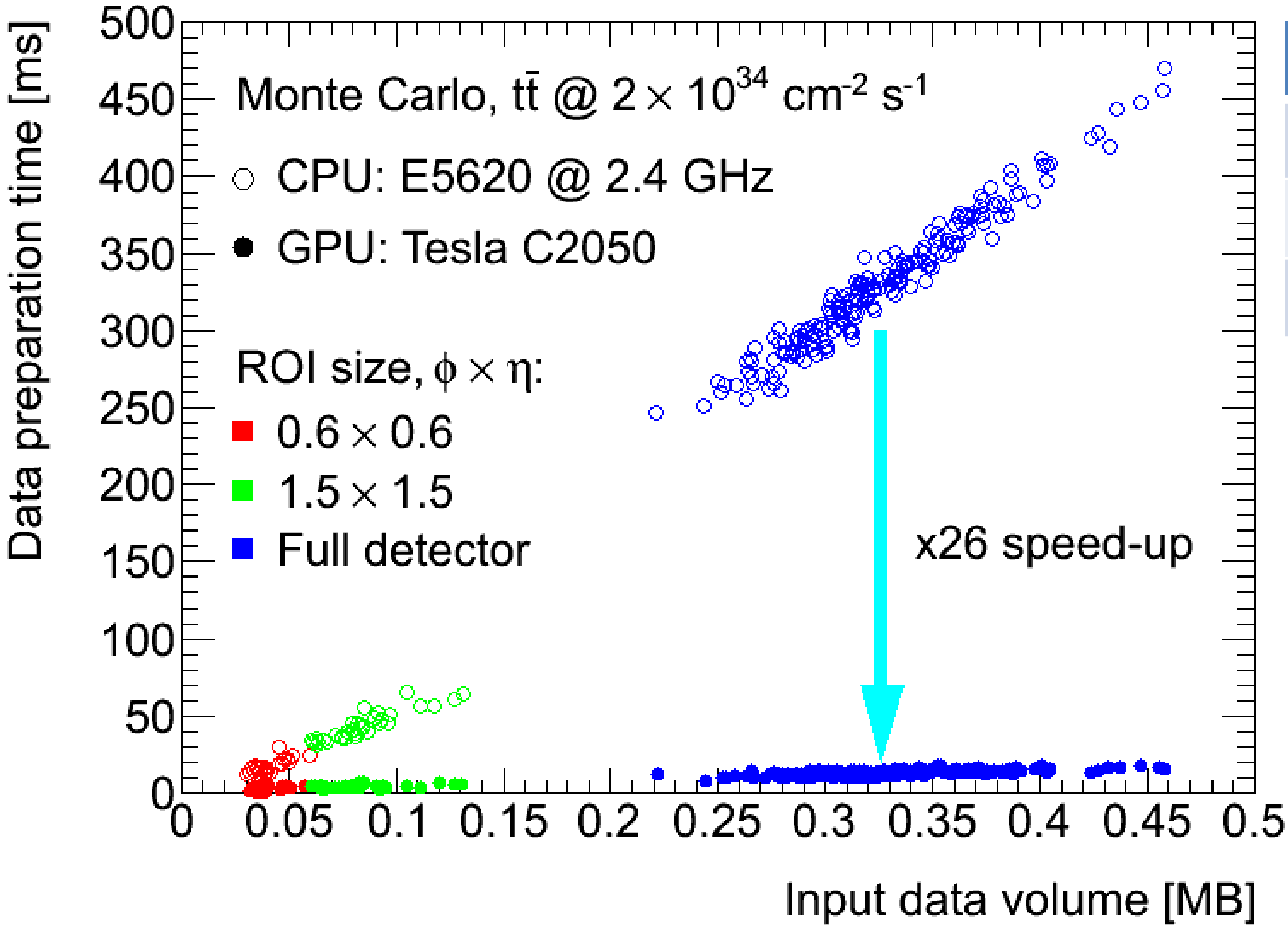
Coprocessors for HLT (GPUs, Intel MIC) ?

- currently at the level of an R&D project (!)
  - ➡ track reconstruction obvious candidate for such an architecture
- interesting proposal is client-server architecture
  - ➡ GPU coprocessor servers
    - algorithms delegate CPU intensive processing
  - ➡ requires messaging layer
    - with support in framework
  - ➡ possible for our HLT farm
    - not obvious on the GRID



- prototype testbed
  - ➡ fully functional Level-2 tracking chain with GPU versions of
    - data preparation: raw data decoding and cluster finding
    - GPU version of Level-2 track finding (without clone removal)
  - ➡ permits to do timing studies

Data preparation: GPU vs. CPU



RoI type	Speed-up
Tau 0.6x0.6	9
B-phys 1.5x1.5	12
FullScan	26

Full data preparation from Bytestream to spacepoints in Pixel and SCT takes

- 3ms for Tau RoI
- only 12 ms for FullScan

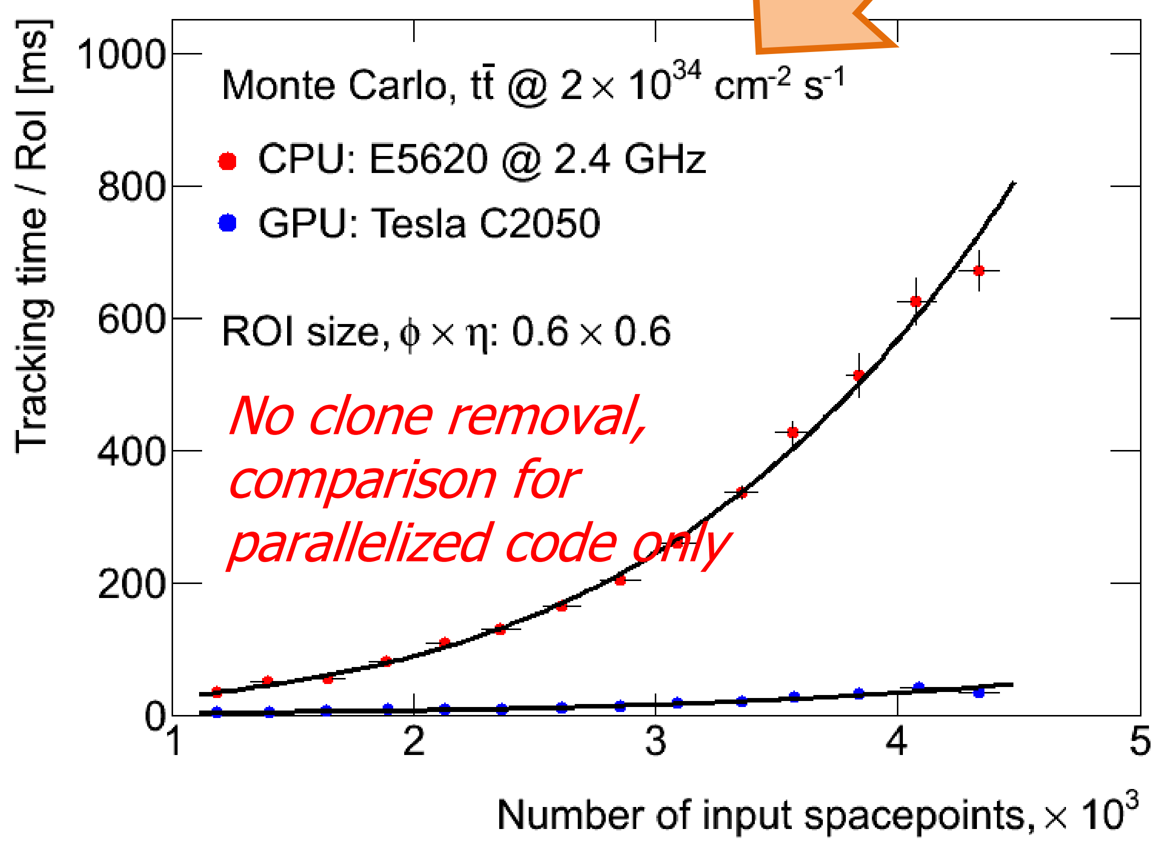
- "GPU-to-CPU" cluster copy test, FullScan, Pixel clusters only:

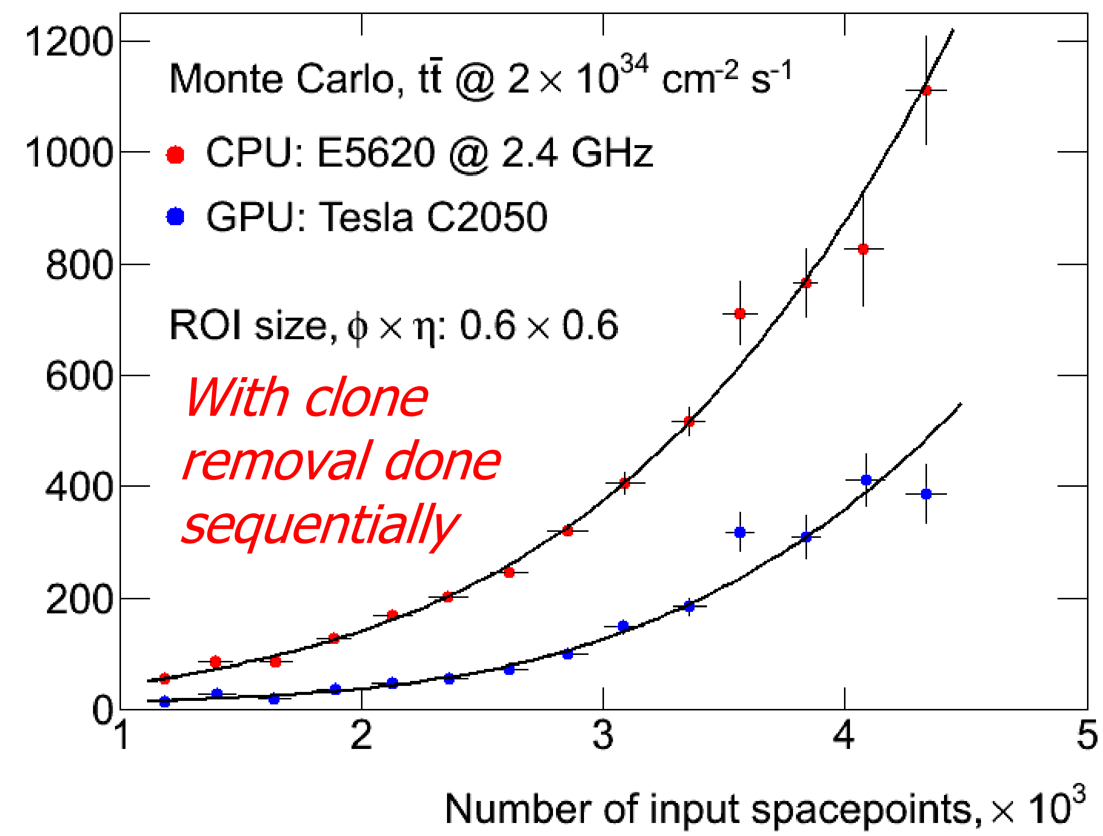
Stage	Production on GPU	Data transfer	Fill RDO and RIO IDCs	IDCs clean-up
Time, ms	6	2	14	8

GPU-accelerated track finding

- LVL2-only “tauNoCut” chain, comparison with TrigSiTrack-00-07-11

SiTrack stage	Seeding	Seed extension	Triplet merging	Clone removal
CPU	8.3	155.6	7.4	70.0
GPU	1.6	7.8	3.4	70.0

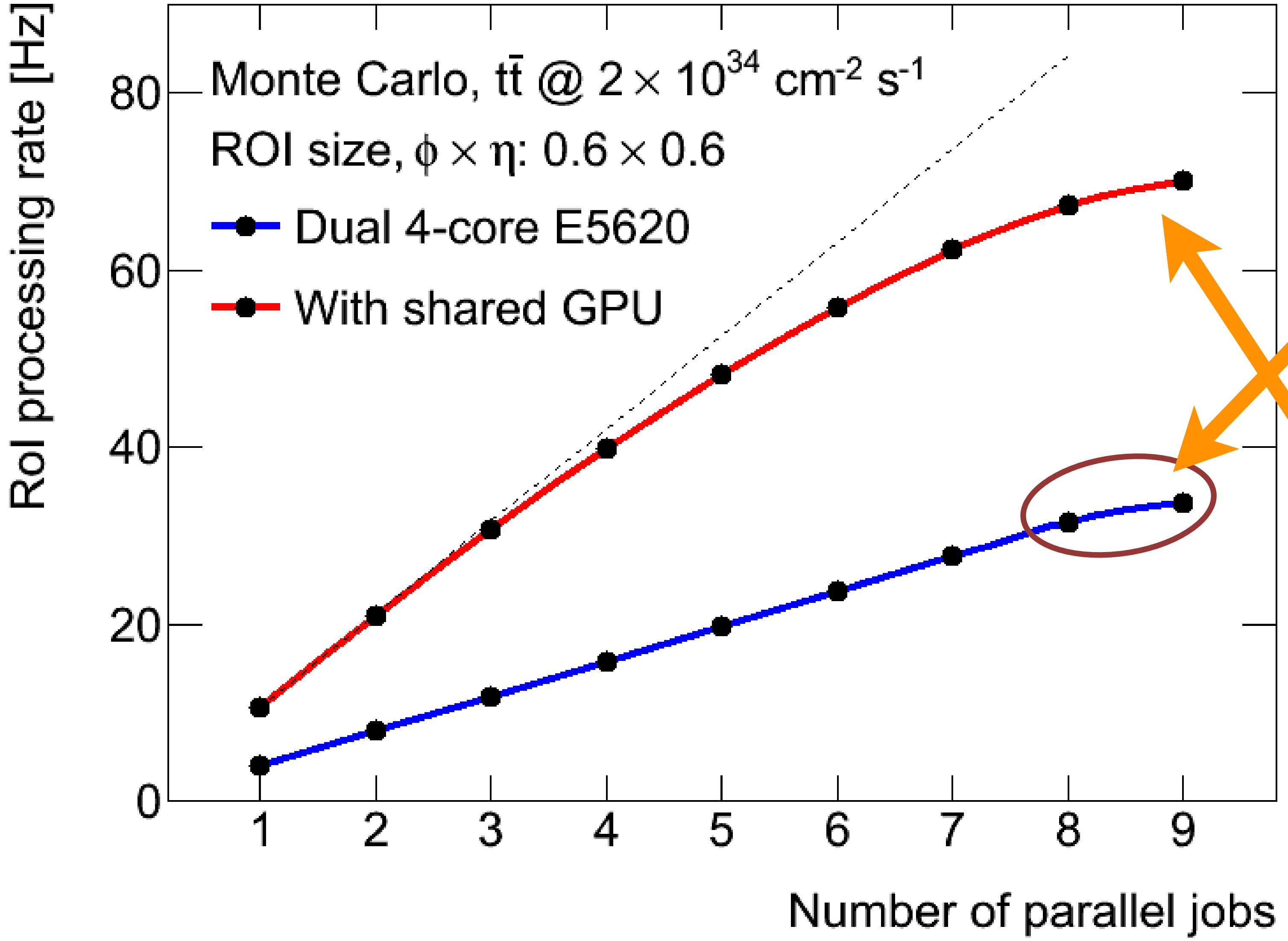




- Parallel code runs 12 times faster, but the overall speed-up is only ~3 due to the sequential clone removal – Amdahl’s Law in action !

GPU sharing test

- LVL2-only "tauNoCut" chain, data preparation and tracking done on GPU



Processor(s)	Rate
Single CPU core	3.9
8 cores + GPU	70.0

effect of hyper-threading ?

The GPU rate saturation needs further studies:

- it could be due to the GPU server process interference with another server or Athena process on the same core

- For this test, 8 cores + GPU are equivalent to ~18 cores running one job per core

Summary

- gave an overview over the future of Inner Detector track reconstruction in ATLAS
- support for Inner Detector upgrade program
- LS1 software updates to deal with technical performance with ever increasing levels of high pileup
- Integrated Simulation Framework will lead to a significant simulation speedup
- discussed new developments in the Trigger tracking