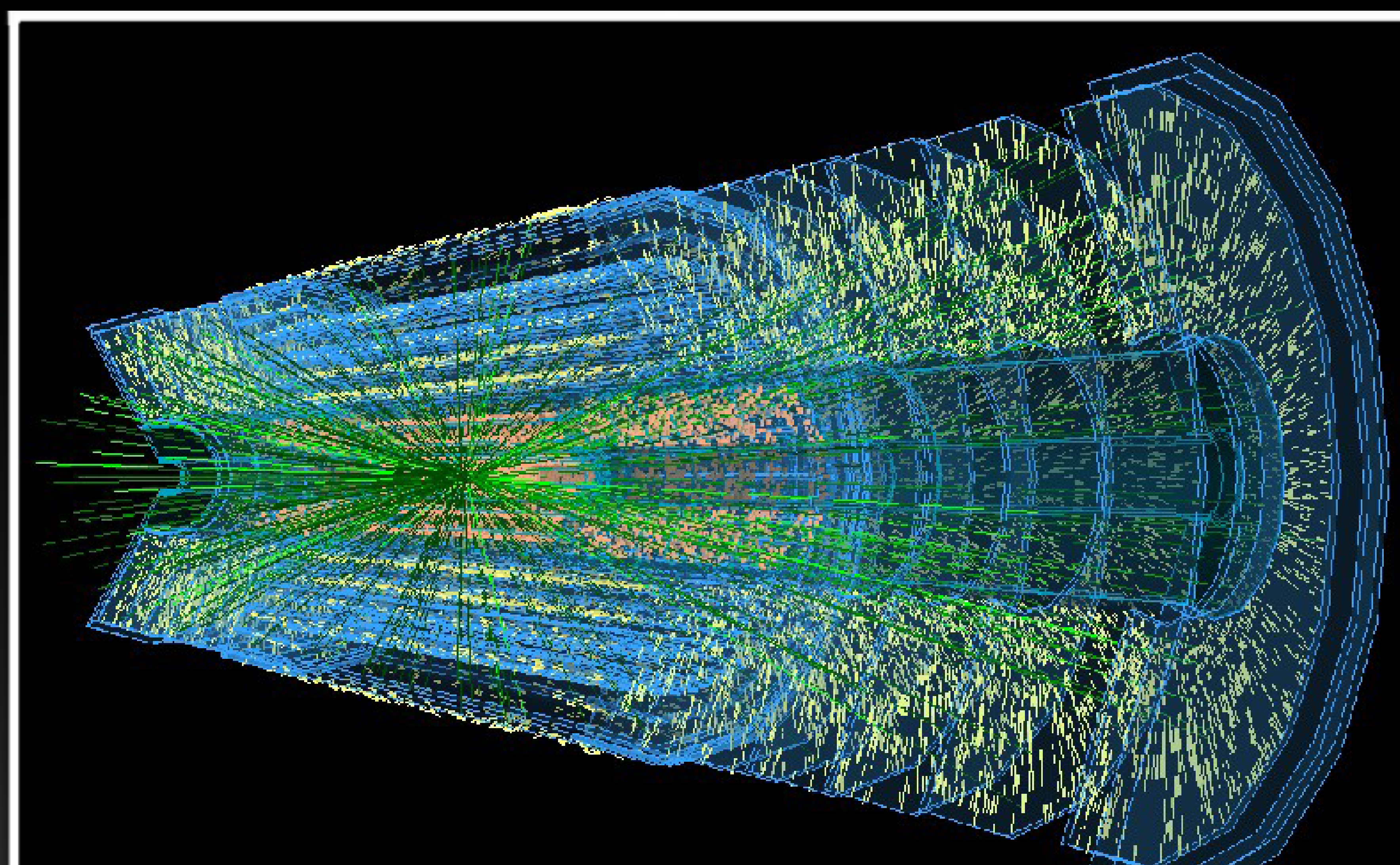
Tracking for High Pileup

Markus Elsing

Seminar at University Wuppertal, July 3rd, 2014



ATLAS HL-LHC event in new tracker

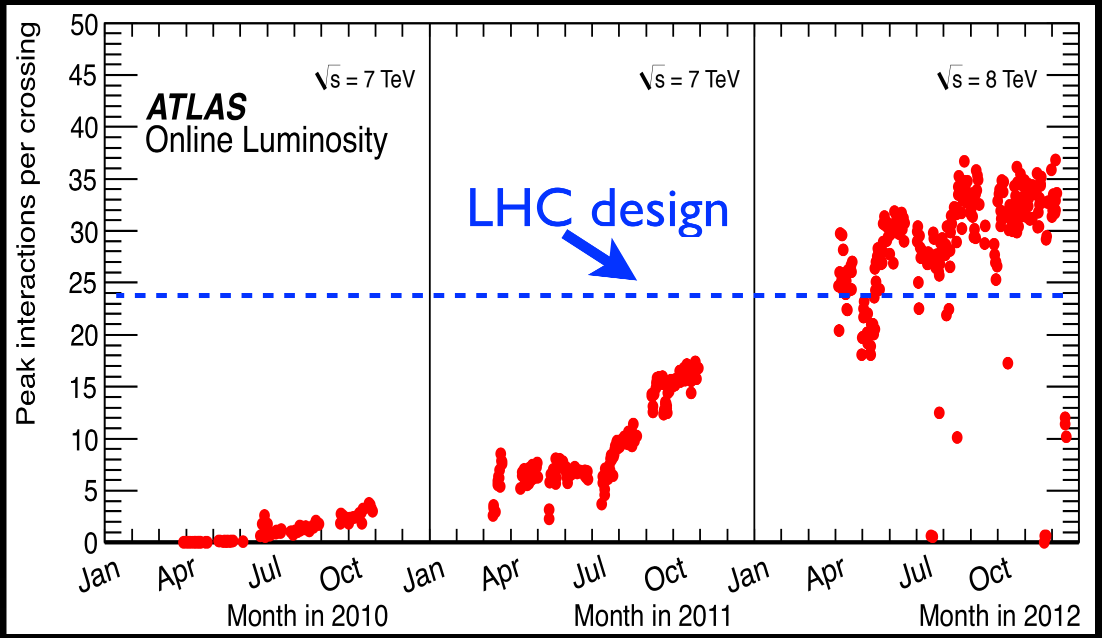
Outline

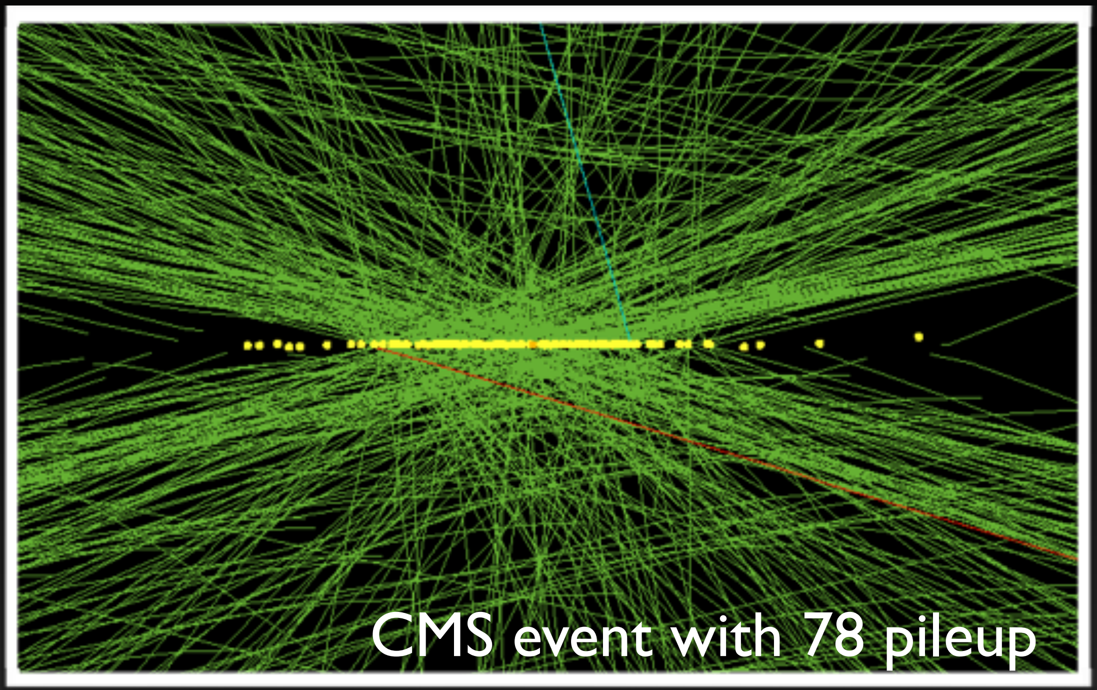
- Introduction: the Challenge
- The present Detectors and Reconstruction Strategies
- Preparing for Run-2 in current Long Shutdown (LS-1)
- What is coming next ?
- Tracking on Many Core Processors
- Tracking and Detector Upgrades
- New Ideas for Track Reconstruction ?
- Conclusions ...

Introduction: the Challenge

Experience with Pileup during Run-1

- pileup in 2012 exceeded design
  - ➡ average pileup up to 35 (1.5 × design)
  - ➡ due 50 nsec operation
- good stability of performance
  - ➡ thanks to several algorithmic improvements
    - for pileup levels seen so far
  - ➡ test with high pileup runs look promising
    - known limitations when going much further
- ATLAS / CMS upgrade goals
  - ➡ upgrade both, hardware and software
  - ➡ restore (and if possible, improve on) physics performance at increasing pileup
    - and stay within computing resources
  - ➡ includes major upgrades of the tracking detectors in view of the pileup at HL-LHC



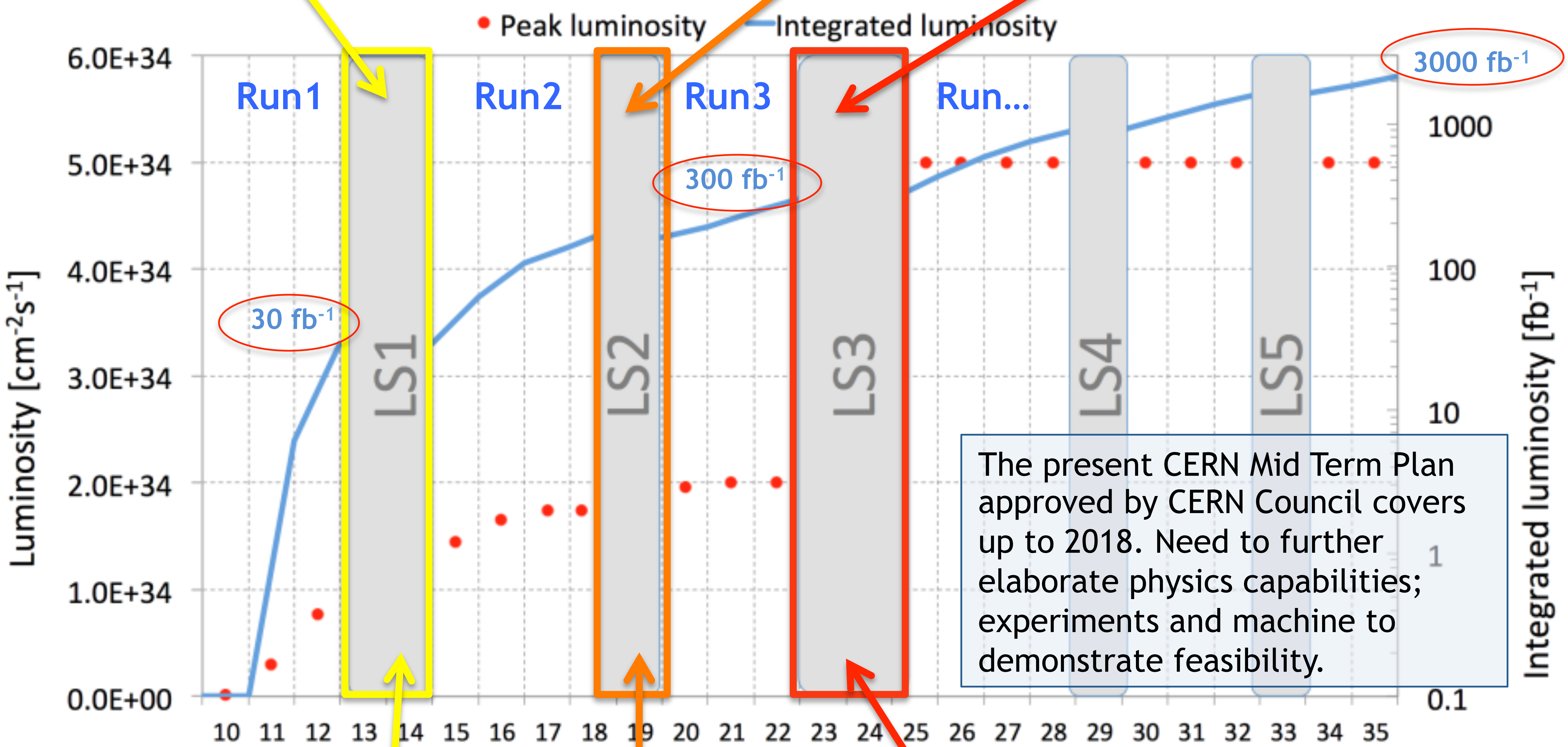


LHC schedule

Fix interconnects and overcome energy limitation

Injector upgrade for high intensity, low emittance bunches, collimation, cryogenics

HL-LHC: Major intervention on 1.2 km of LHC



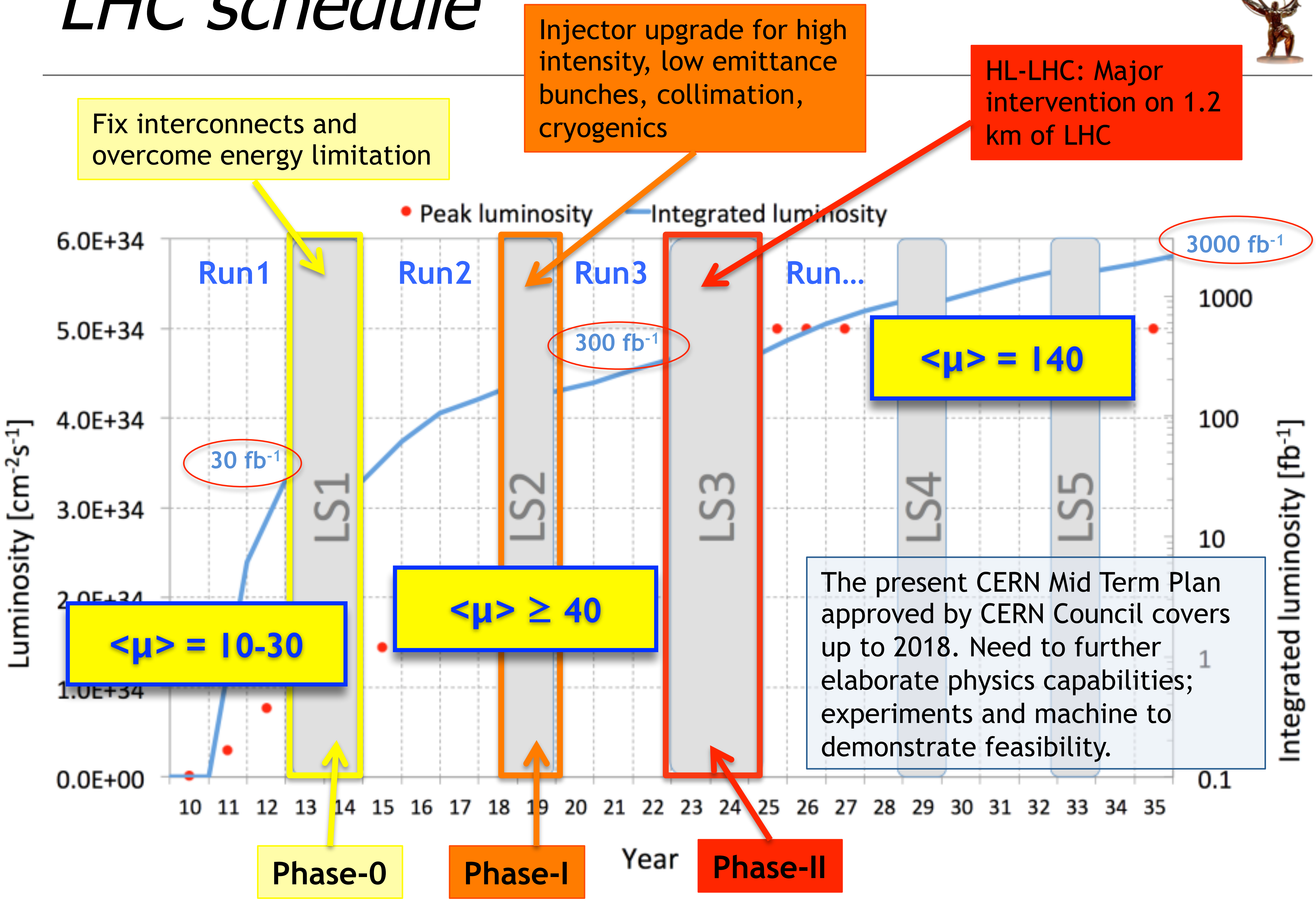
The present CERN Mid Term Plan approved by CERN Council covers up to 2018. Need to further elaborate physics capabilities; experiments and machine to demonstrate feasibility.

Phase-0 **Phase-I** **Phase-II**

LHC schedule approved by CERN management and LHC experiments spokespersons and technical coordinators (December 2013)

C.Gemme, LHCP

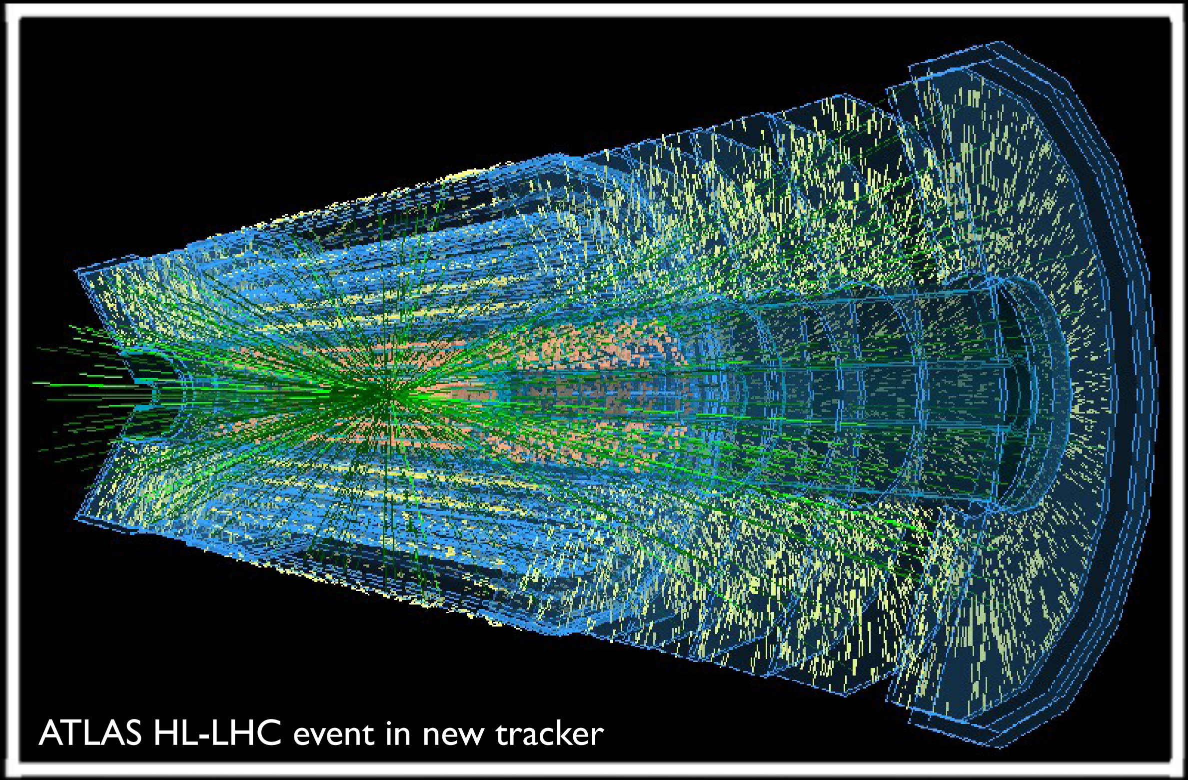
LHC schedule



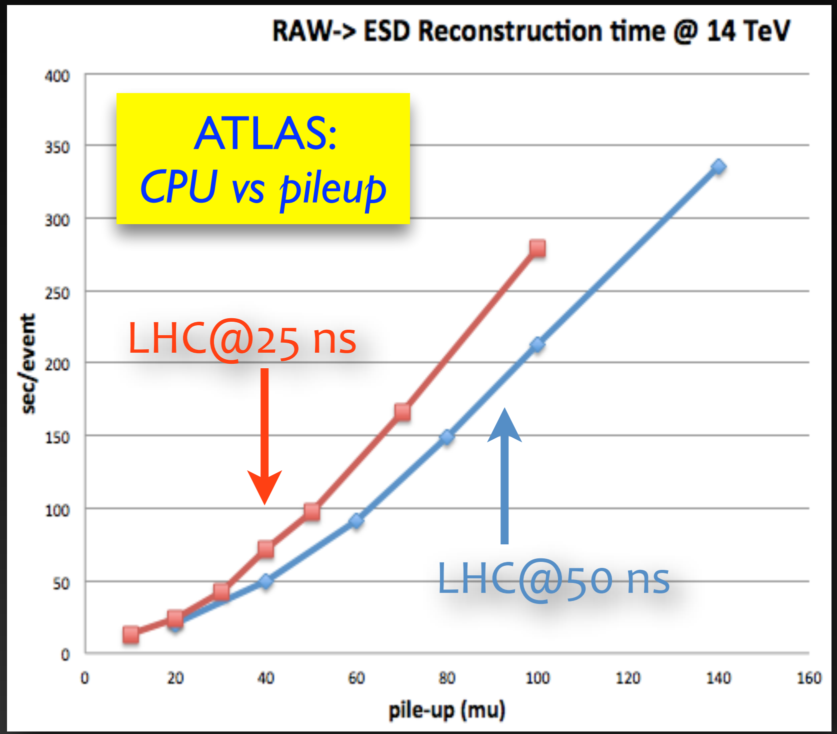
LHC schedule approved by CERN management and LHC experiments spokespersons and technical coordinators (December 2013)

C.Gemme, LHCP

Tracking at HL-LHC ?


ATLAS HL-LHC event in new tracker

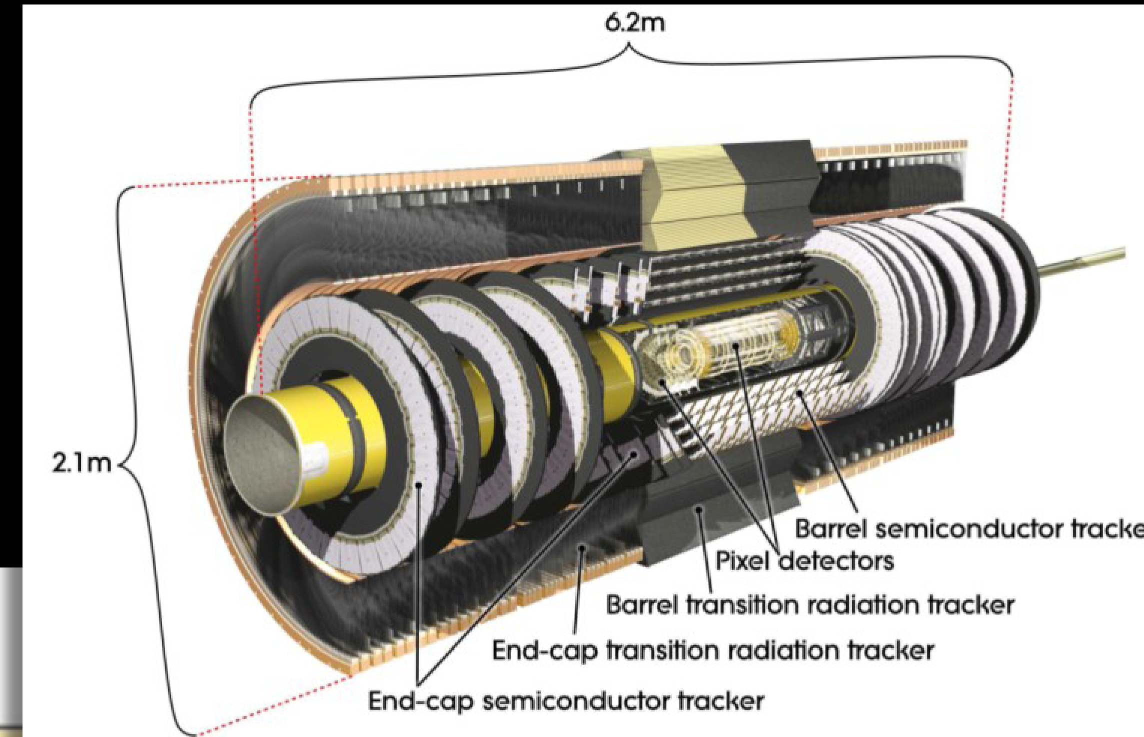
- track reconstruction
  - ➡ combinatorics grows with pileup
  - ➡ naturally resource driver (CPU/memory)
- the million dollar question:
  - ➡ how to reconstruct LH-LHC events within resources ? (pileup ~ 140-200)
- this is not a new question !
  - ➡ we knew that tracking at the LHC is going to be a problem
    - hence: we aim at improving over something that is highly optimised
  - ➡ processor technologies are changing as well
    - need to rethink some of the design decisions we did
    - will require vectorisation and multi-threading
    - improve data locality (avoid cache misses), etc.

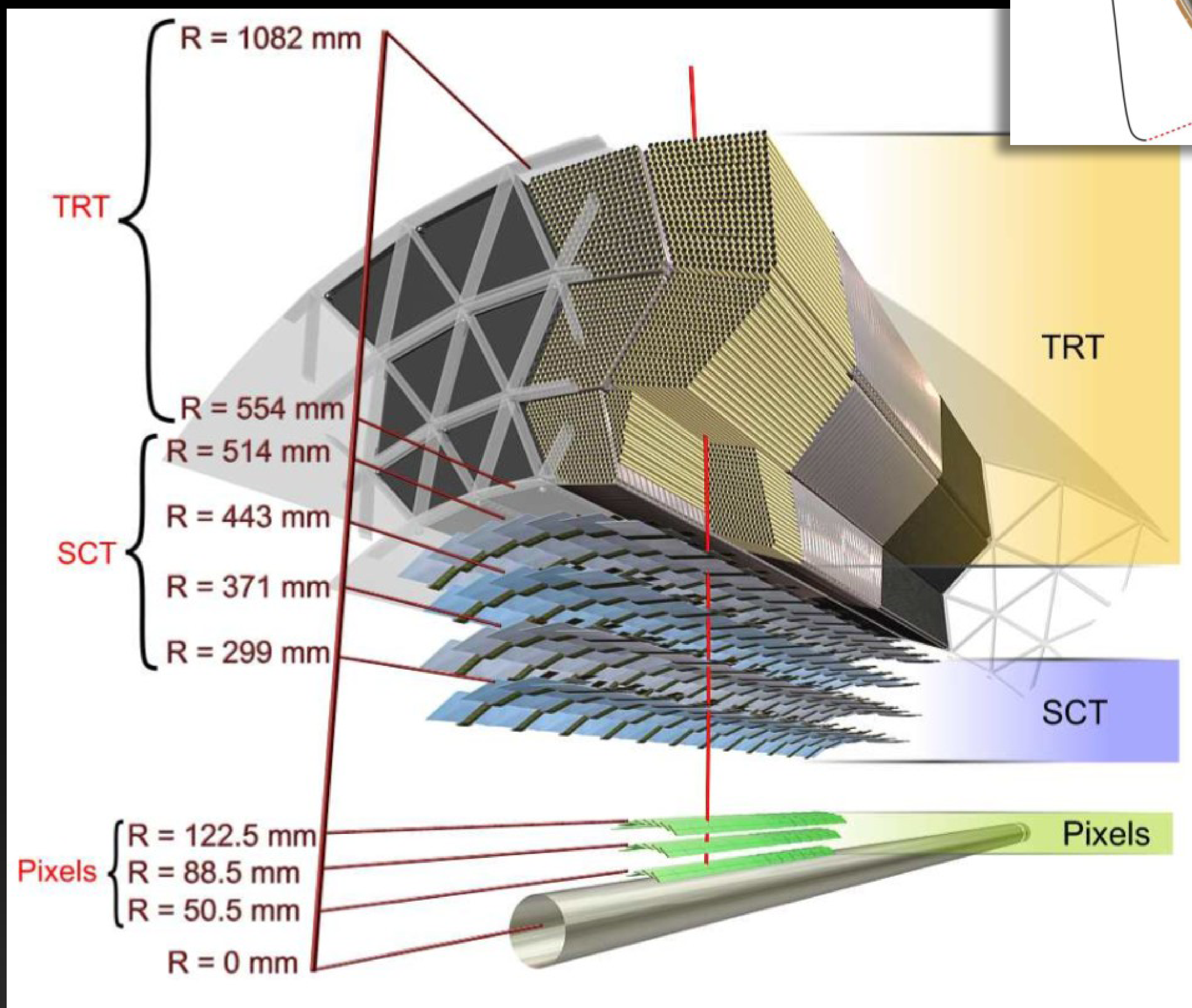


The present Detectors and Reconstruction Strategies

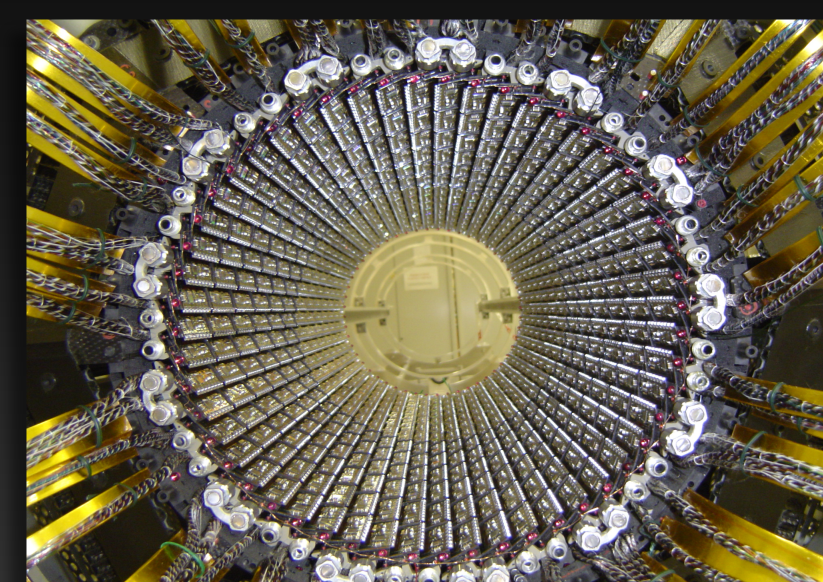
ATLAS Inner Detector

- optimised for 24 pileup events





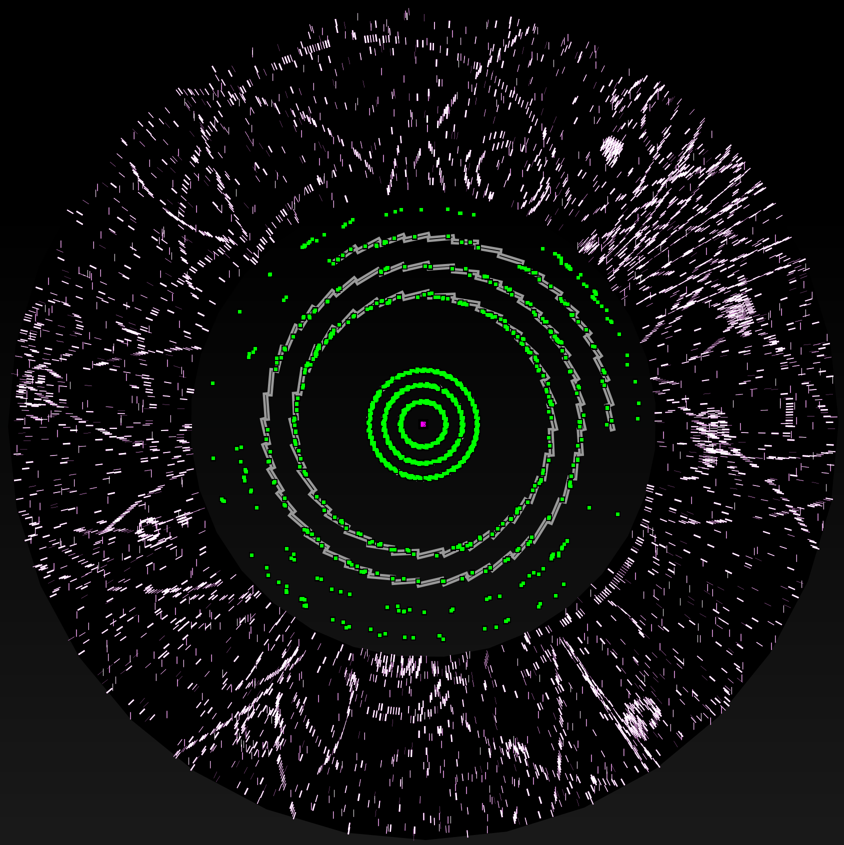
- barrel track passes:
  - ➡ ~36 TRT 4mm straws
  - ➡ 4x2 Si strips on stereo modules12cm x 80 mm, 285mm thick
  - ➡ 3 pixel layers, 250mm thick



Introduction: NewTracking in ATLAS

pre-precessing

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



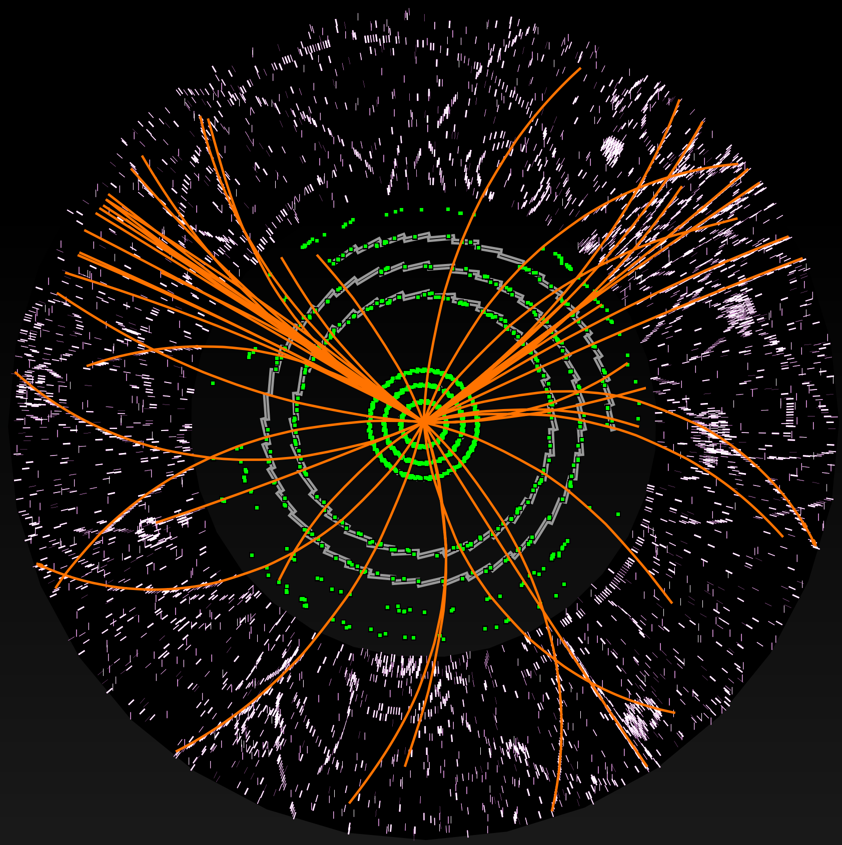
Introduction: NewTracking in ATLAS

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



Introduction: NewTracking in ATLAS

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...

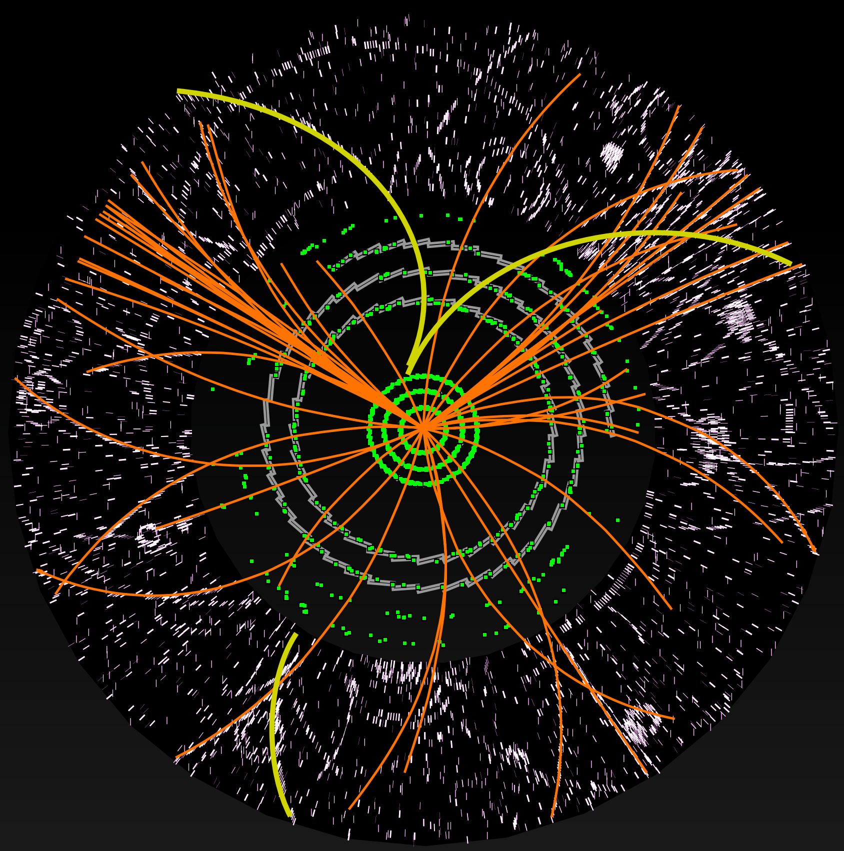
extension into TRT
- progressive finder
- refit of track and selection

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



Introduction: NewTracking in ATLAS

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

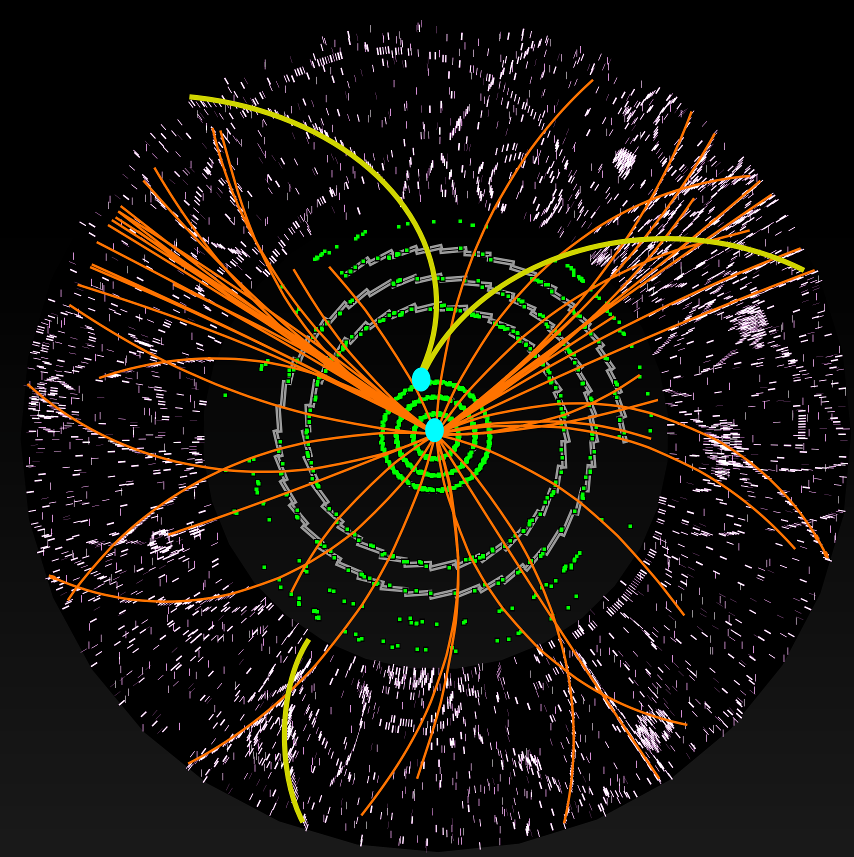
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

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



Introduction: NewTracking in ATLAS

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

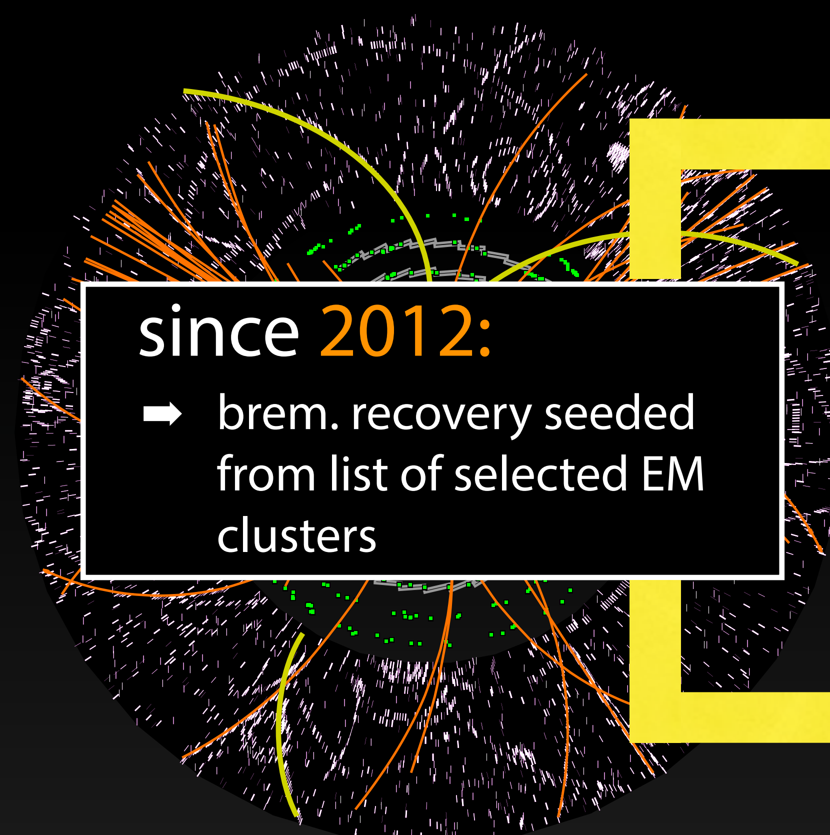
TRT segment finder
- ➡ on remaining drift circles
- ➡ uses Hough transform

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

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

standalone TRT
- ➡ unused TRT segments

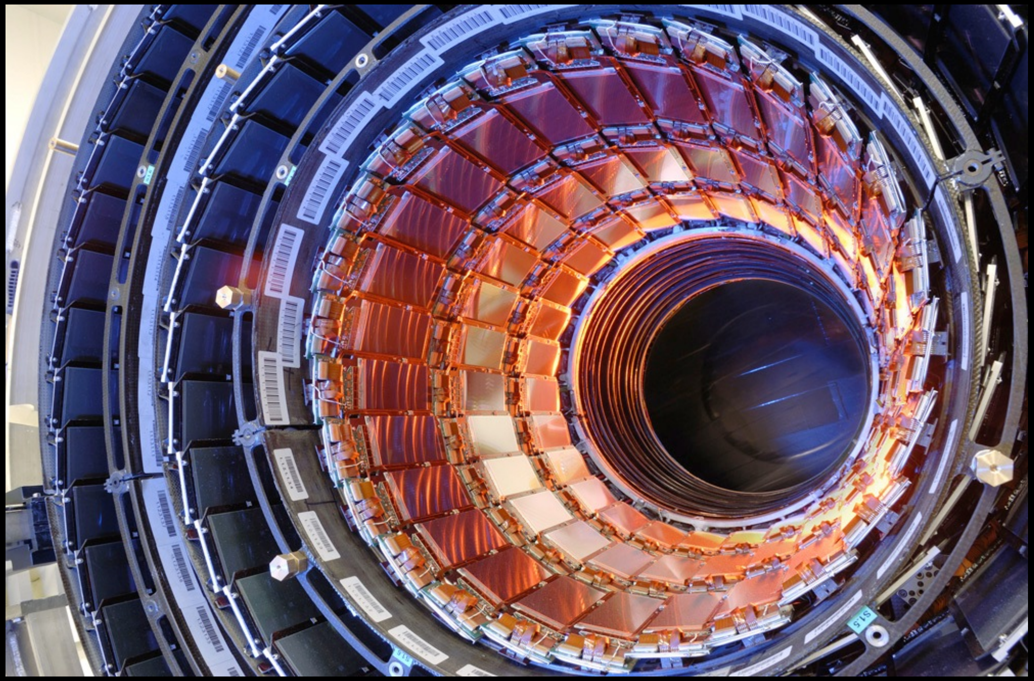
vertexing
- ➡ primary vertexing
- ➡ conversion and V0 search

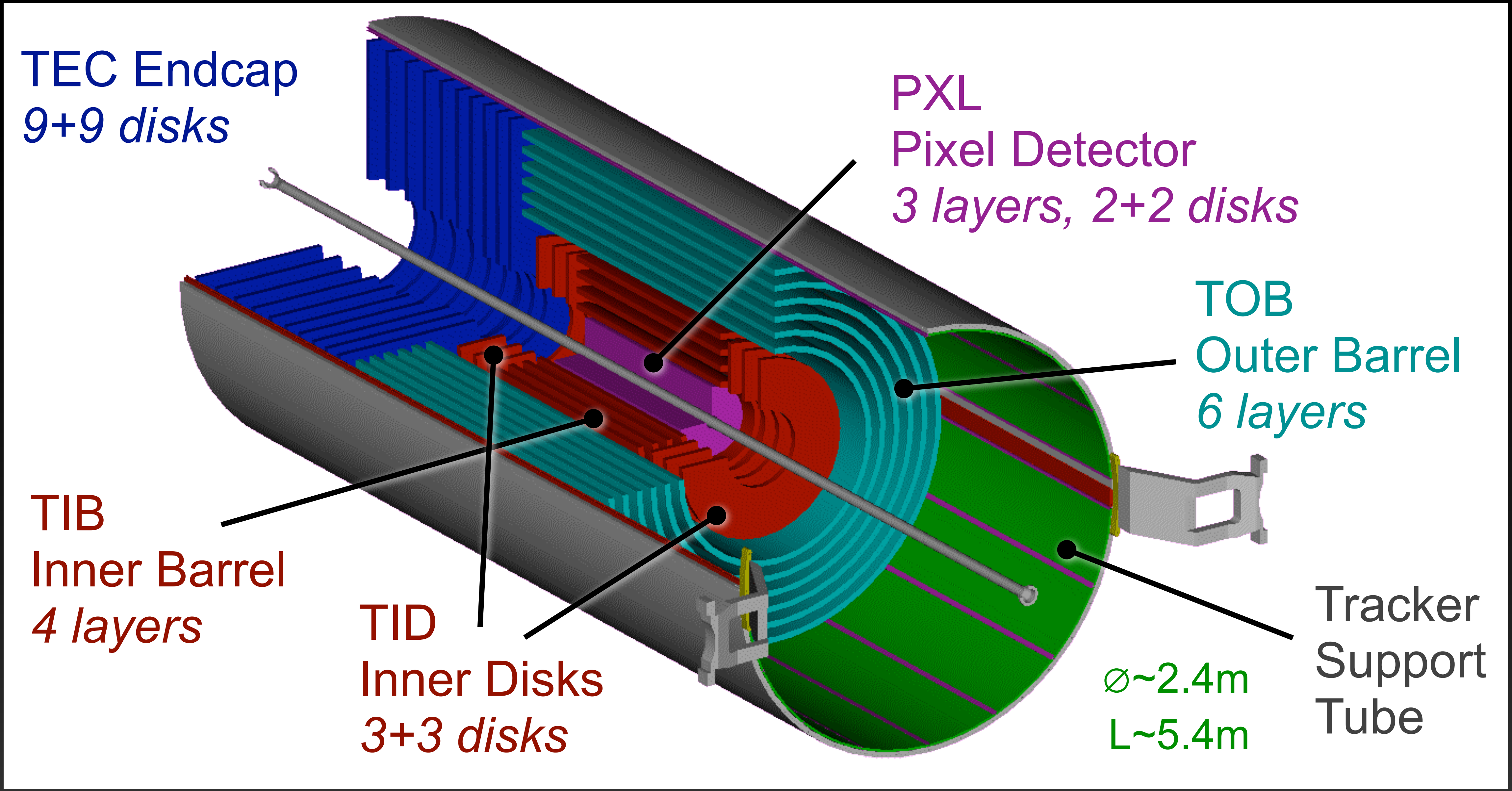


since 2012:
- ➡ brem. recovery seeded from list of selected EM clusters

CMS Tracker

- largest silicon tracker ever built
  - ➡ **Pixels:** 66M channels, 100x150 μm² Pixel
  - ➡ **Si-Strip detector:** ~23m³, 210m² of Si area, 10.7M channels

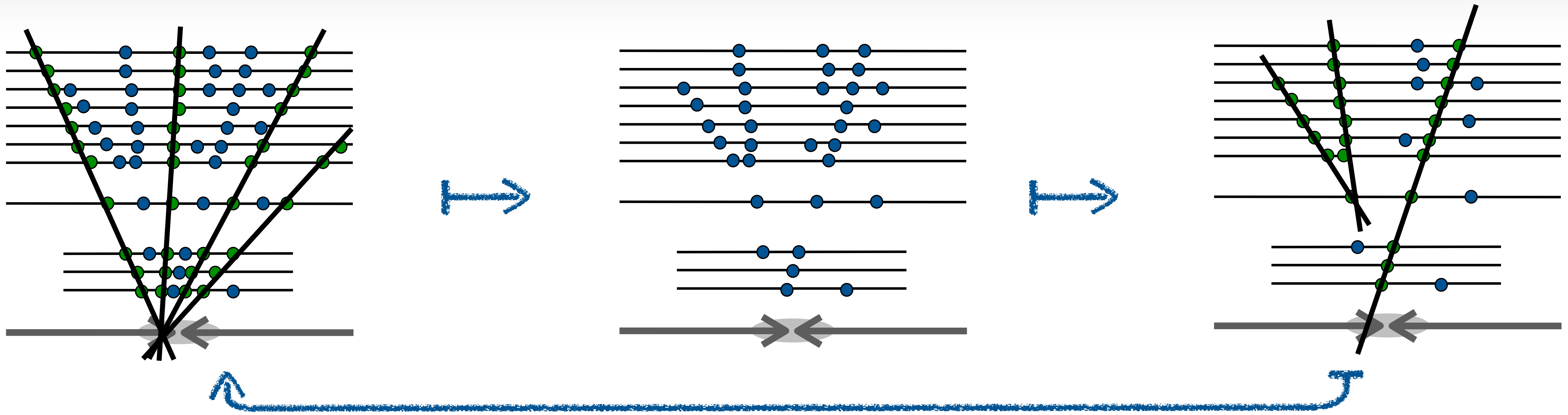




Sguazzoni et al.,

GSI Tracking Workshop 2012

Iterative tracking



The CMS tracking relies on iterations (*steps*) of the tracking procedure; each step works on the remaining not-yet-associated hits and is optimized with respect to the seeding topology and to the final quality cuts.

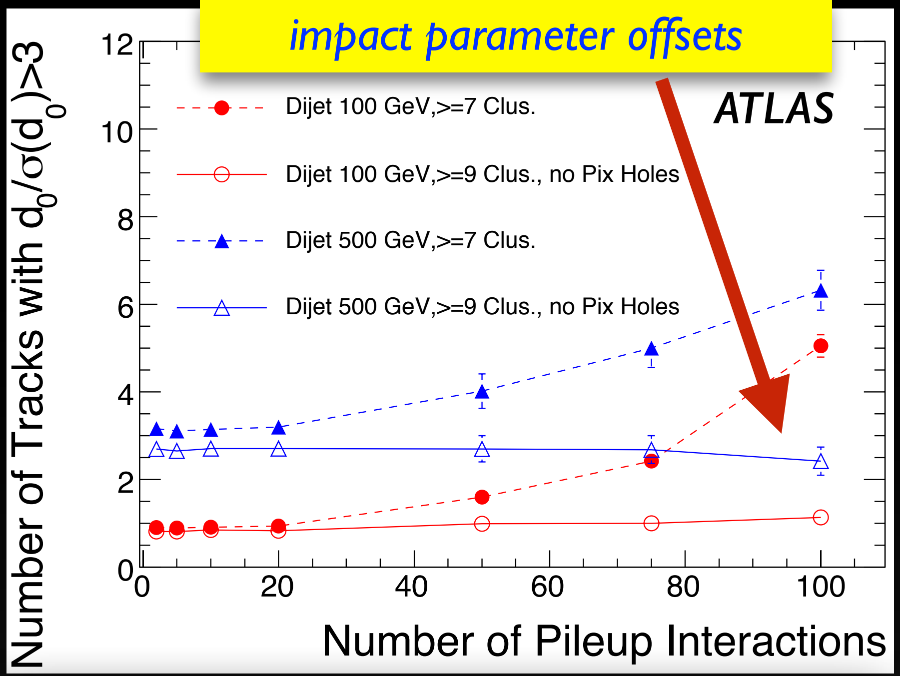
#step	seed type	seed subdetectors	$P_T^{\min}$ [GeV/c]	d_0 cut	z_0 cut
0	triplet	pixel	0.6	**0.02** cm	4.0σ
1	triplet	pixel	0.2	**0.02** cm	4.0σ
2	pair	pixel	0.6	**0.015** cm	0.09 cm
3	triplet	pixel	**0.3**	**1.5** cm	**2.5**σ
4	triplet	pixel/TIB/TID/TEC	**0.5-0.6**	**1.5** cm	10.0 cm
5	pair	TIB/TID/TEC	0.6	2.0 cm	10.0 cm
6	pair	TOB/TEC	0.6	2.0 cm	30.0 cm

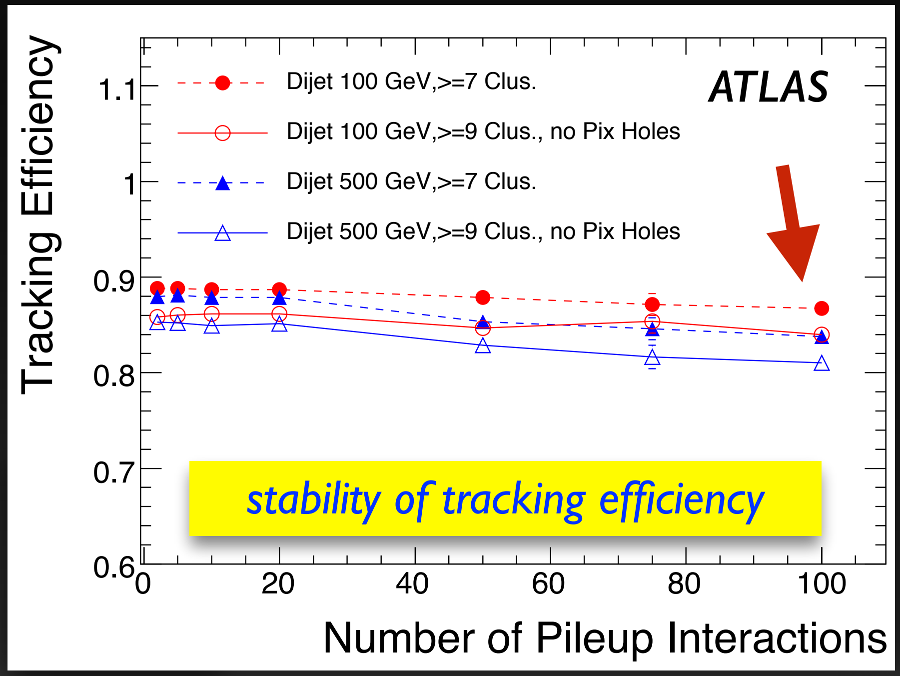
Iterative tracking in 2012 (CMSSW 52x) / In **bold** the changes with respect to 44x

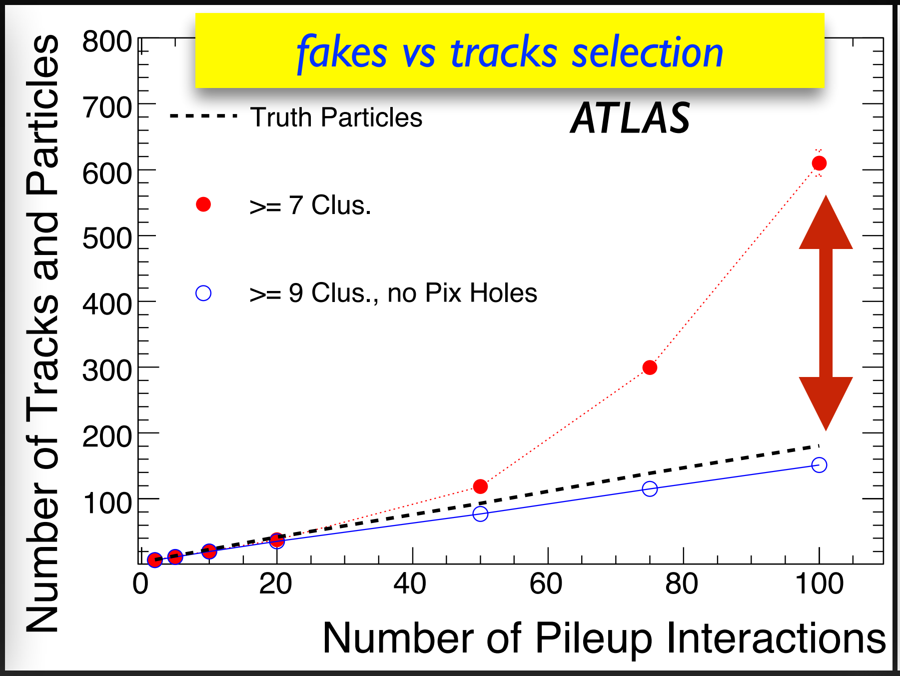
Expected Performance vs Pileup (2008)

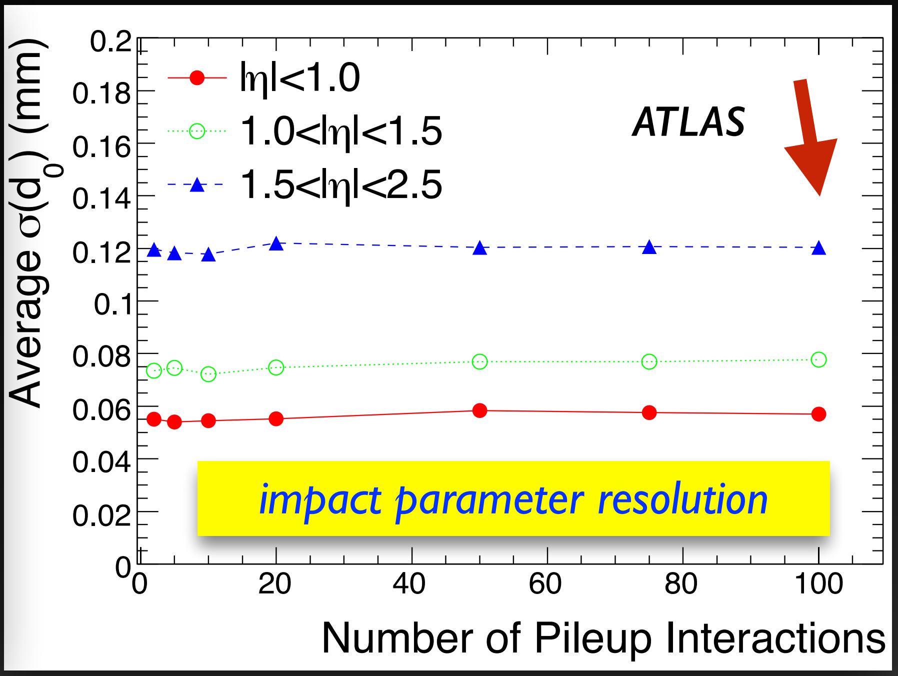
- affects on tracking in current detector
  - ➡ pileup affects physics performance if reconstruction unchanged
    - adjusting track selection allows to mitigate effects
  - ➡ studied extensively even pre-data taking (see plots)

- current tracker ok until ~100 pileup
  - ➡ no effects on efficiencies or resolutions
  - ➡ control fakes and fake impact offsets with tracking cuts
  - ➡ not shown: TRT occupancy effect
    - loss of momentum resolution due to reduced efficiency for precision hits



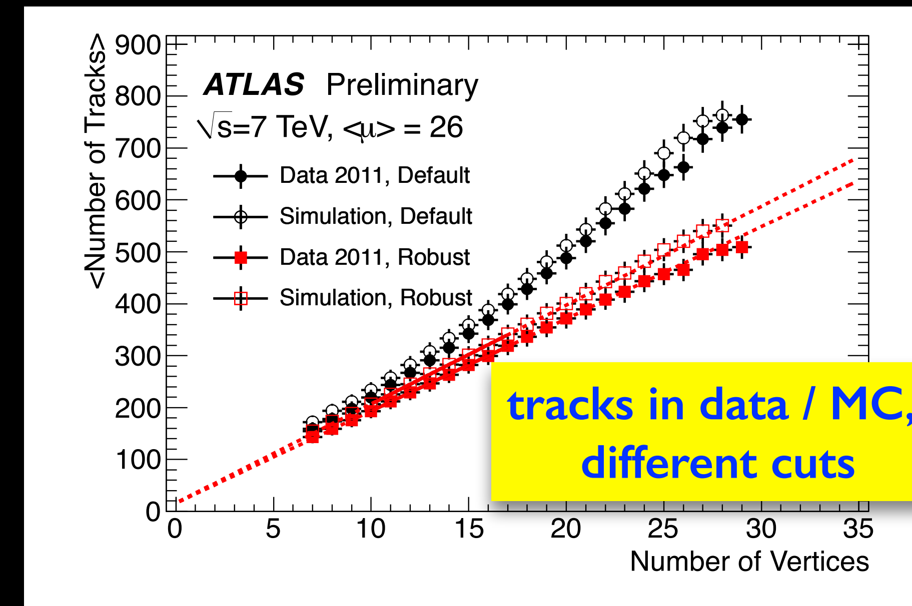


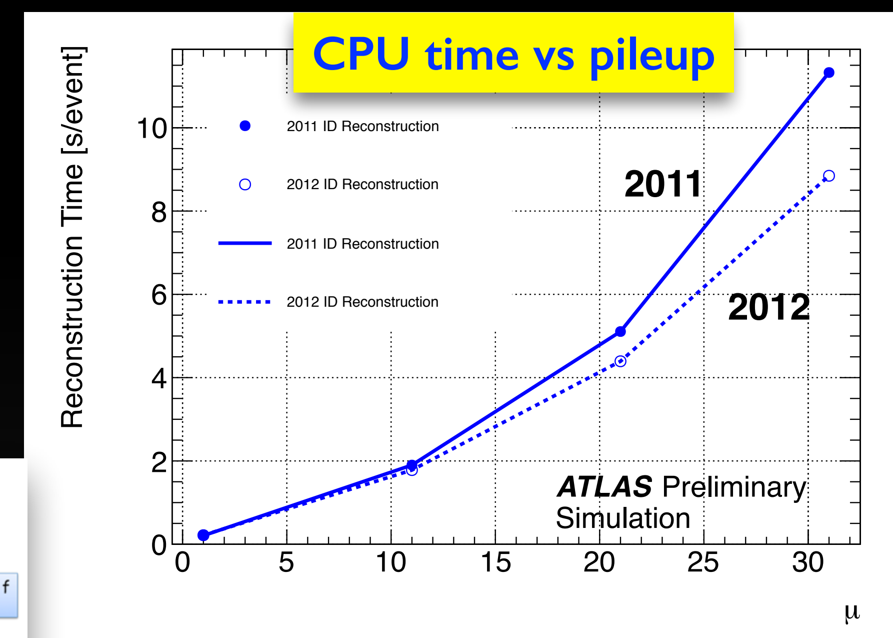


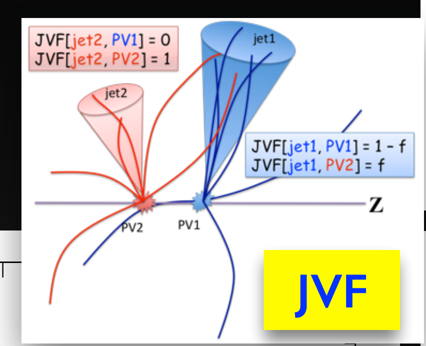


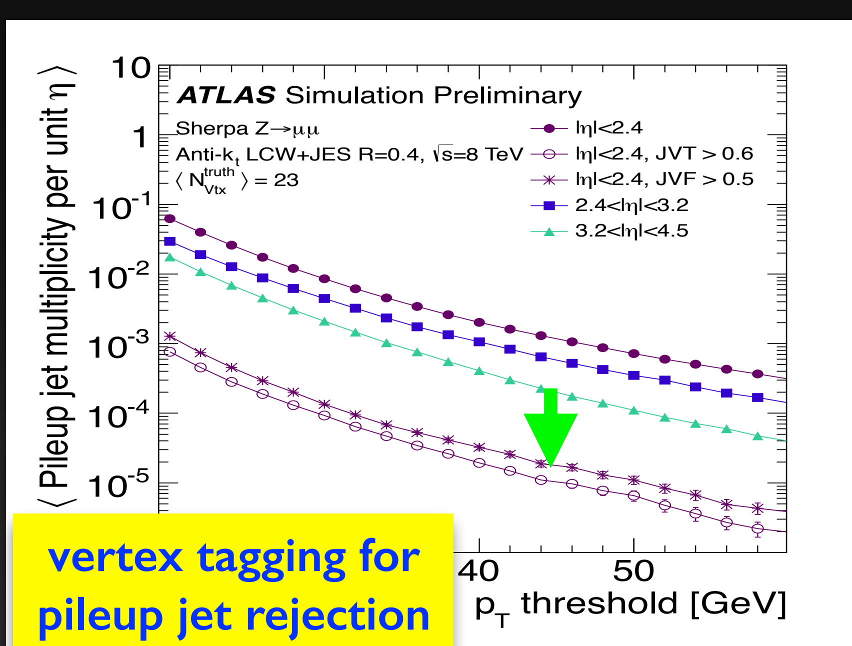
Run-1 Experience with Pileup

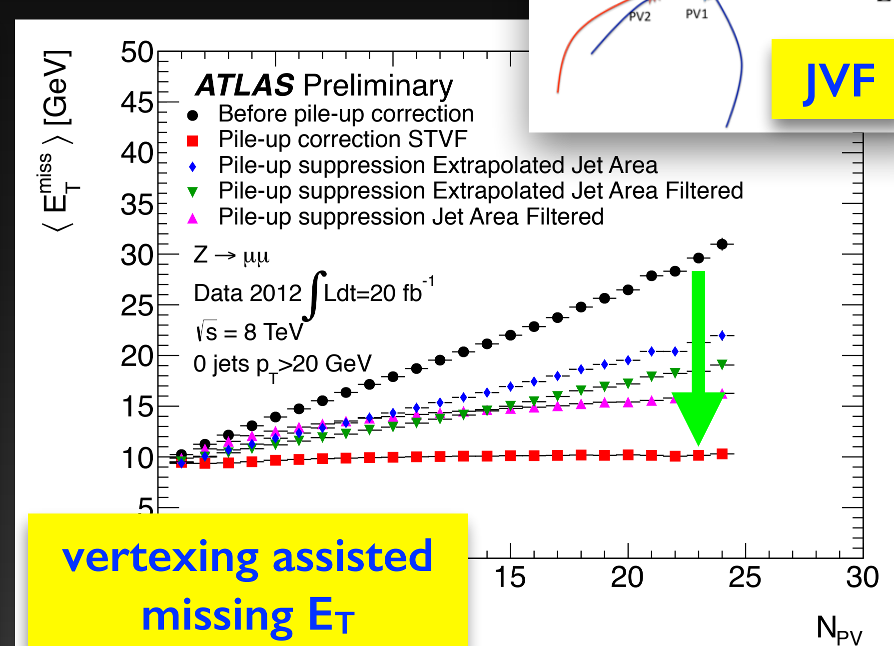
- tracking performance as expected
  - ➡ using more robust tracking cuts controls fakes
  - ➡ CPU increasing rapidly with μ
- primary vertexing
  - ➡ visible effects of vertex merging at high μ
  - ➡ Σp_T based vertex tagging less and less optimal (see MC)
- tracking as a tool for pileup control
  - ➡ jet reconstruction (JVF and variants of it)
  - ➡ ATLAS is developing particle flow

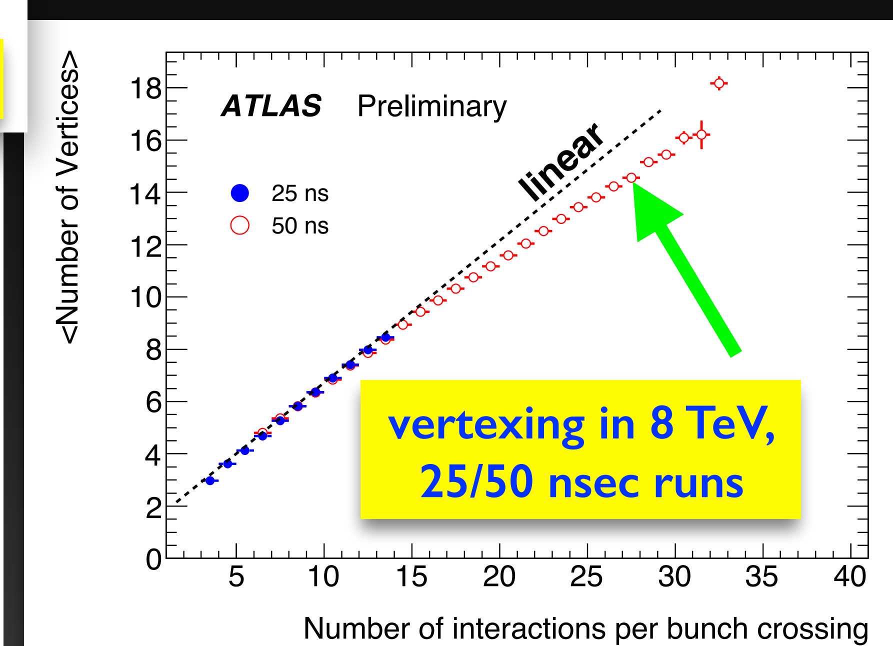








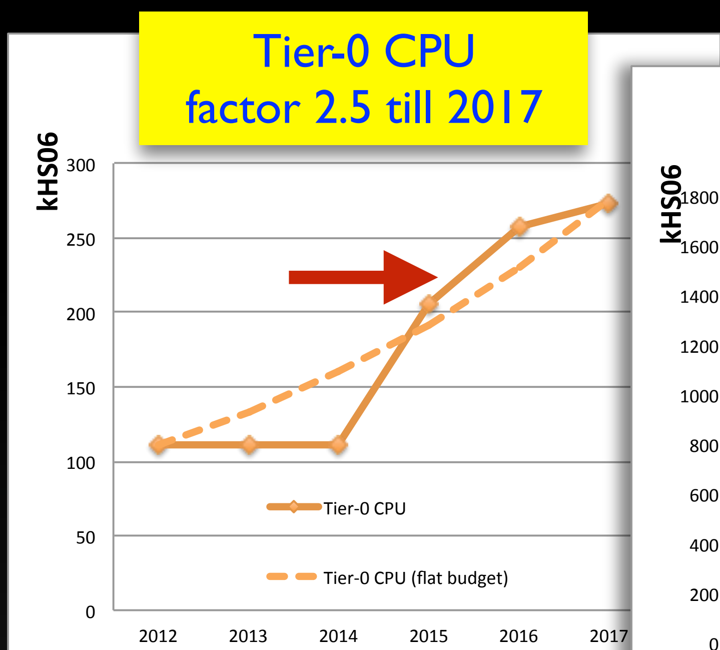


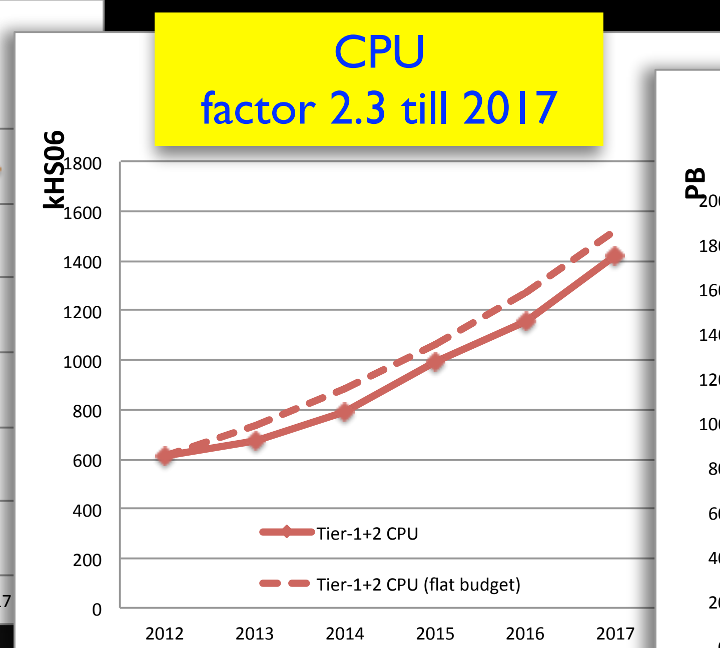


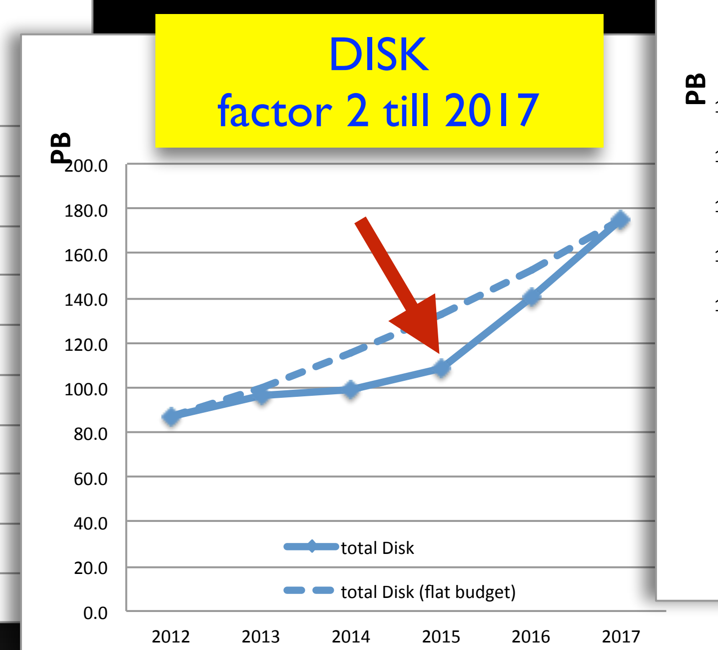
Preparing for Run-2 in current Long Shutdown (LS-1)

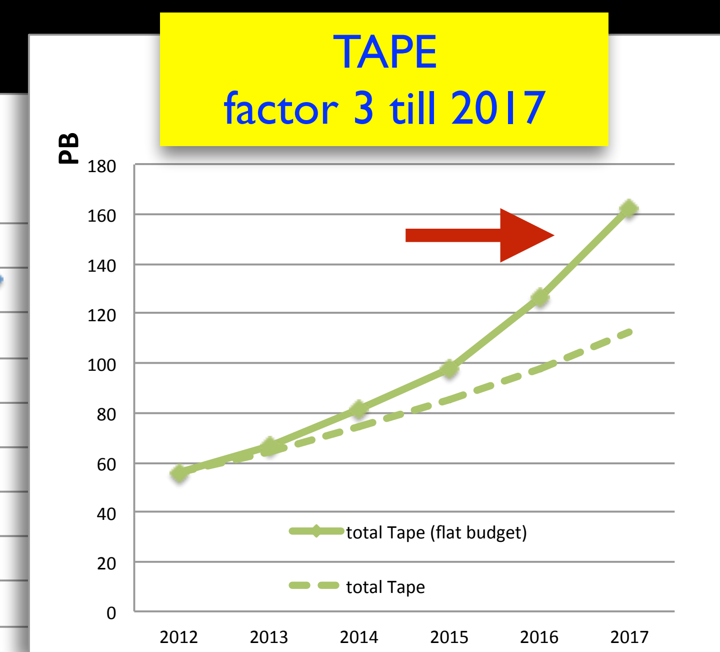
Computing Constraints for Run-2

- unlike Run-1, our computing resources will be limited !
  - ➡ assumption is we stay with a constant computing budget
  - ➡ interplay of technology advancement, market price and needed replacements







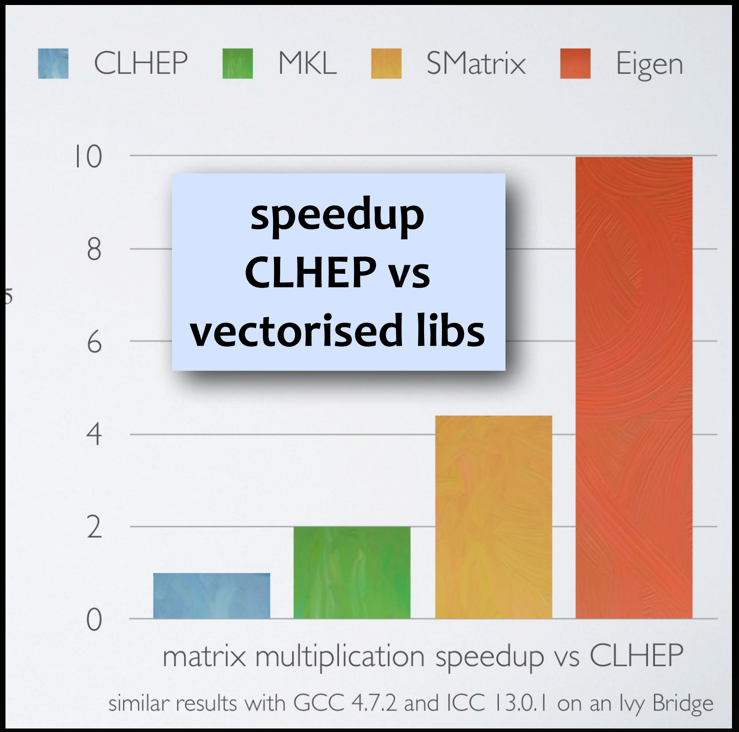


Borut Kersevan, Richard Mount et al.

- motivation for LS1 software upgrades
  - ➡ ensure Tier-0 can process 1kHz trigger rate, required to keep single lepton triggers
  - ➡ optimise disk usage (see new Analysis Model)
  - ➡ "soften" disk and CPU limits on Monte Carlo statistics
- focus here on preparation of tracking for 40 pileup

LS1 Tracking Developments in ATLAS

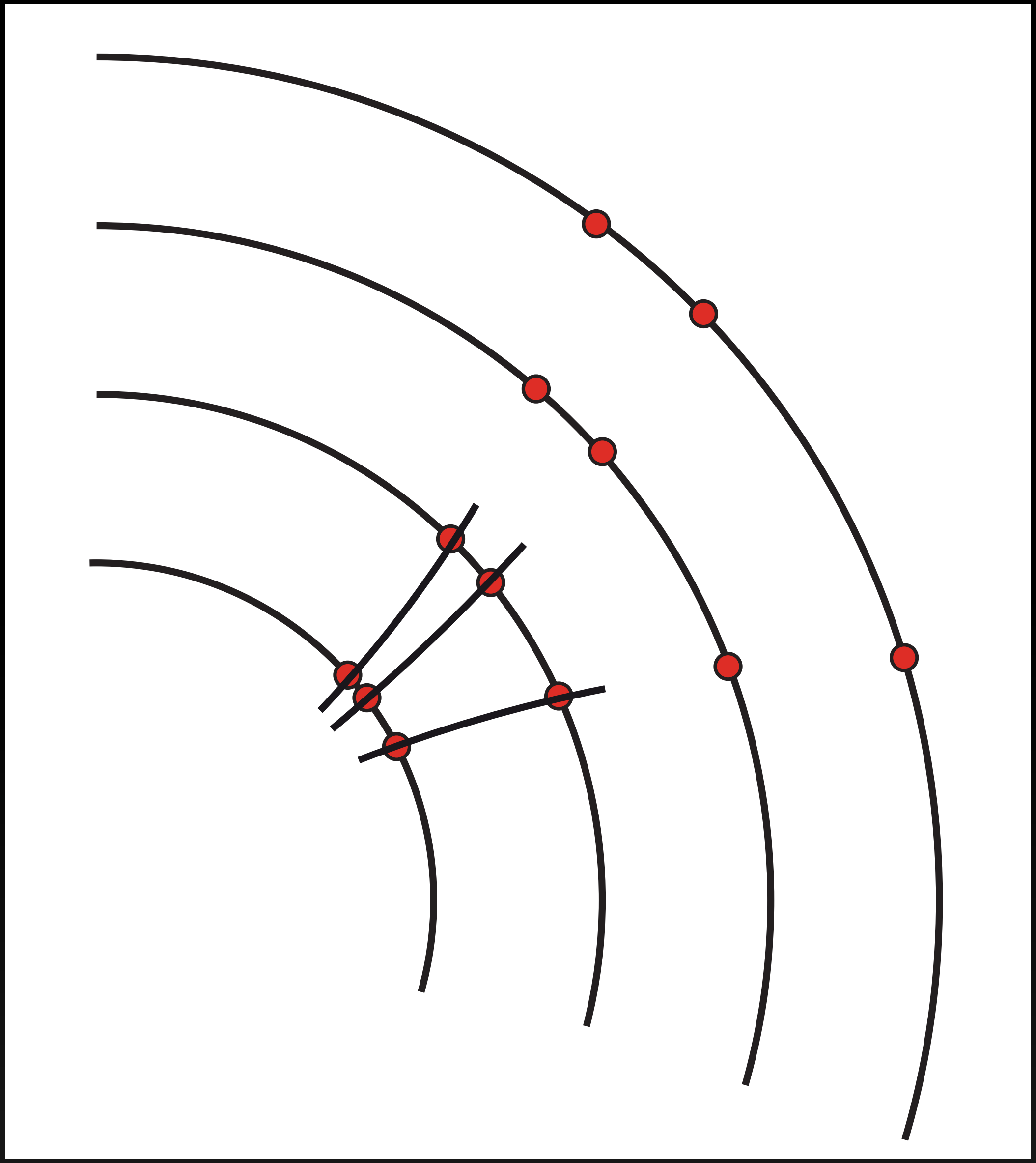
- focus was to work on technology and strategy to improve CURRENT algorithms
  - ➡ technology:
    - simplify EDM design to be less OO ("hip" 10 years ago)
    - Eigen migration - faster vector+matrix algebra
    - vectorised trigonometric functions (VDT, intel math lib)
    - F90 to C++ for the b-field (CPU hot spot)
  - ➡ strategy:
    - work on iterative track finding strategy
    - modified track seeding to explore 4th Pixel layer



- as well...
  - ➡ xAOD: a new analysis EDM
- hence, mix of SIMD and algorithm tuning
  - ➡ further speedups probably requires "new" thinking

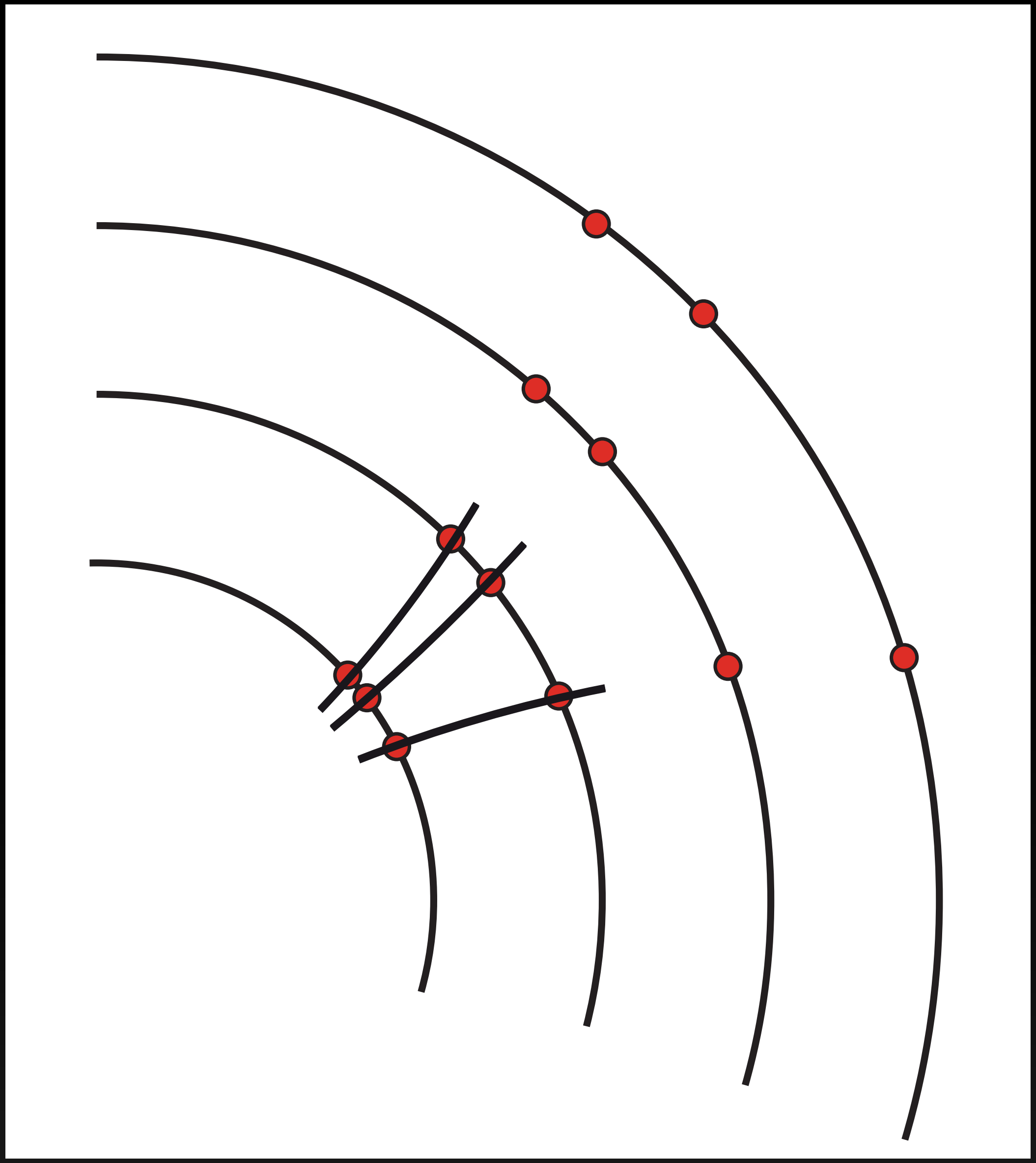
Tuning the Seeding Strategy

- the track finding algorithm



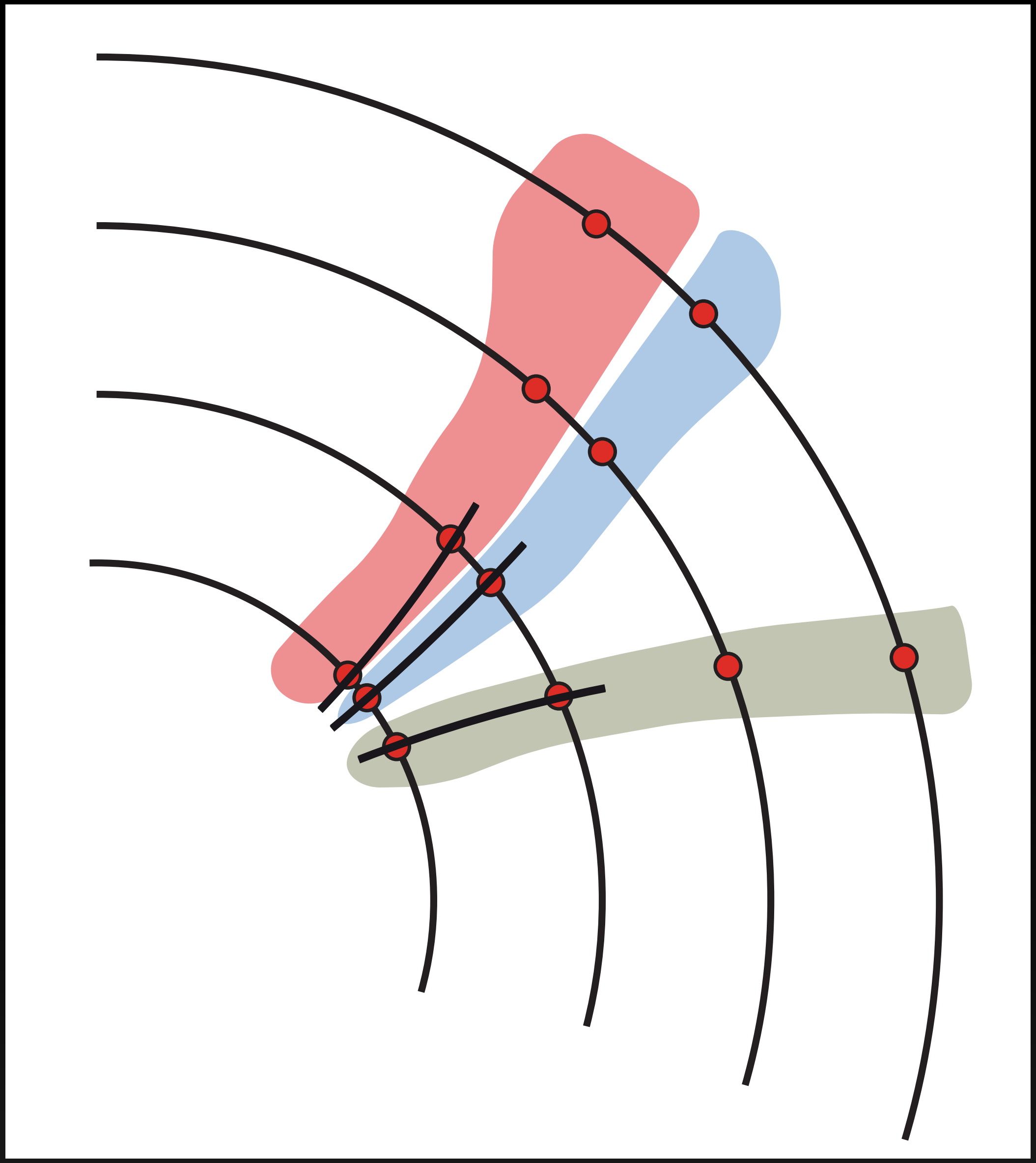
Tuning the Seeding Strategy

- the track finding algorithm
  - ➡ find **seed** from combination of 3 hits
    - search using hough transform



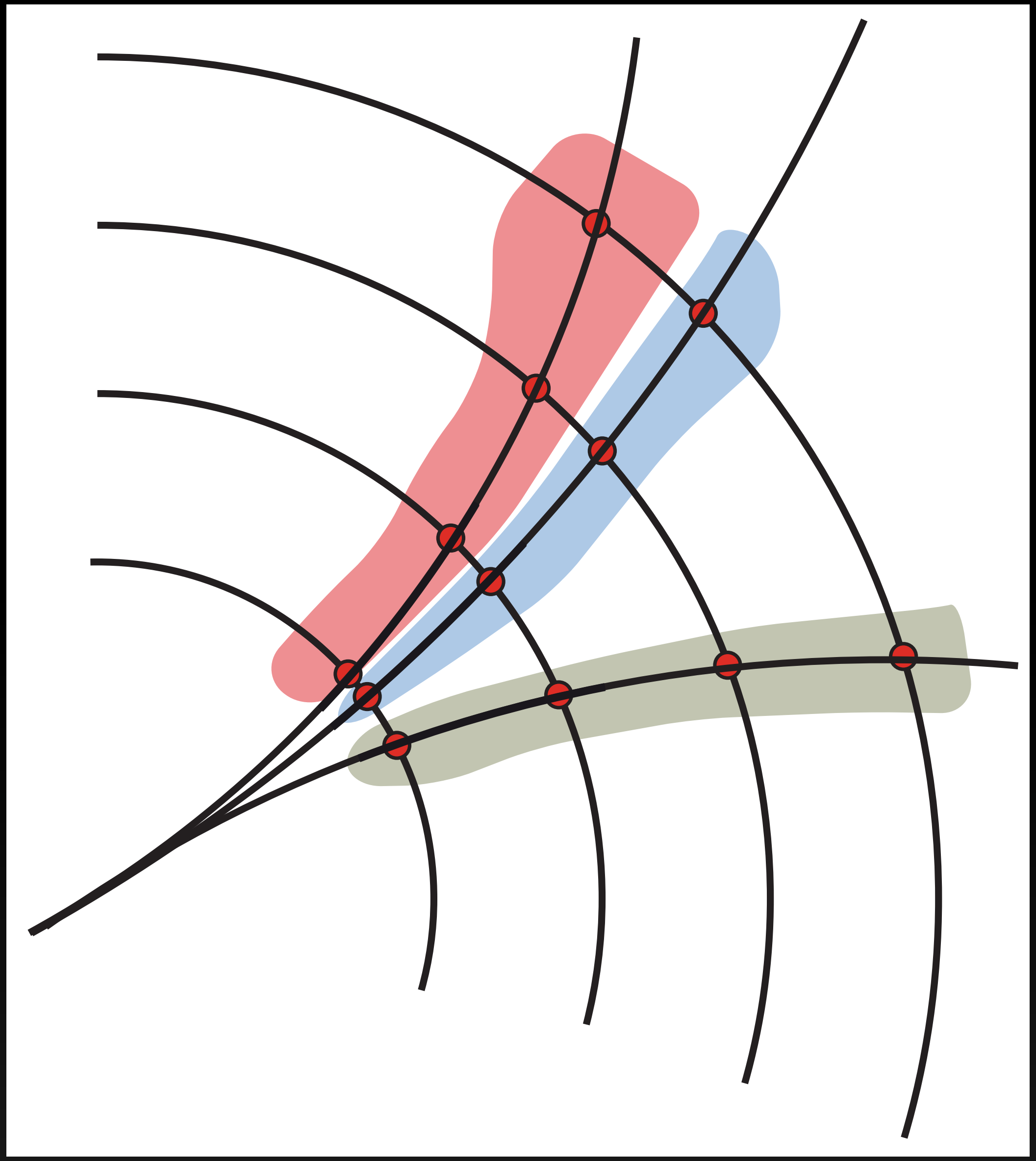
Tuning the Seeding Strategy

- the track finding algorithm
  - ➡ find **seed** from combination of 3 hits
    - search using hough transform
  - ➡ build **road** along the likely trajectory



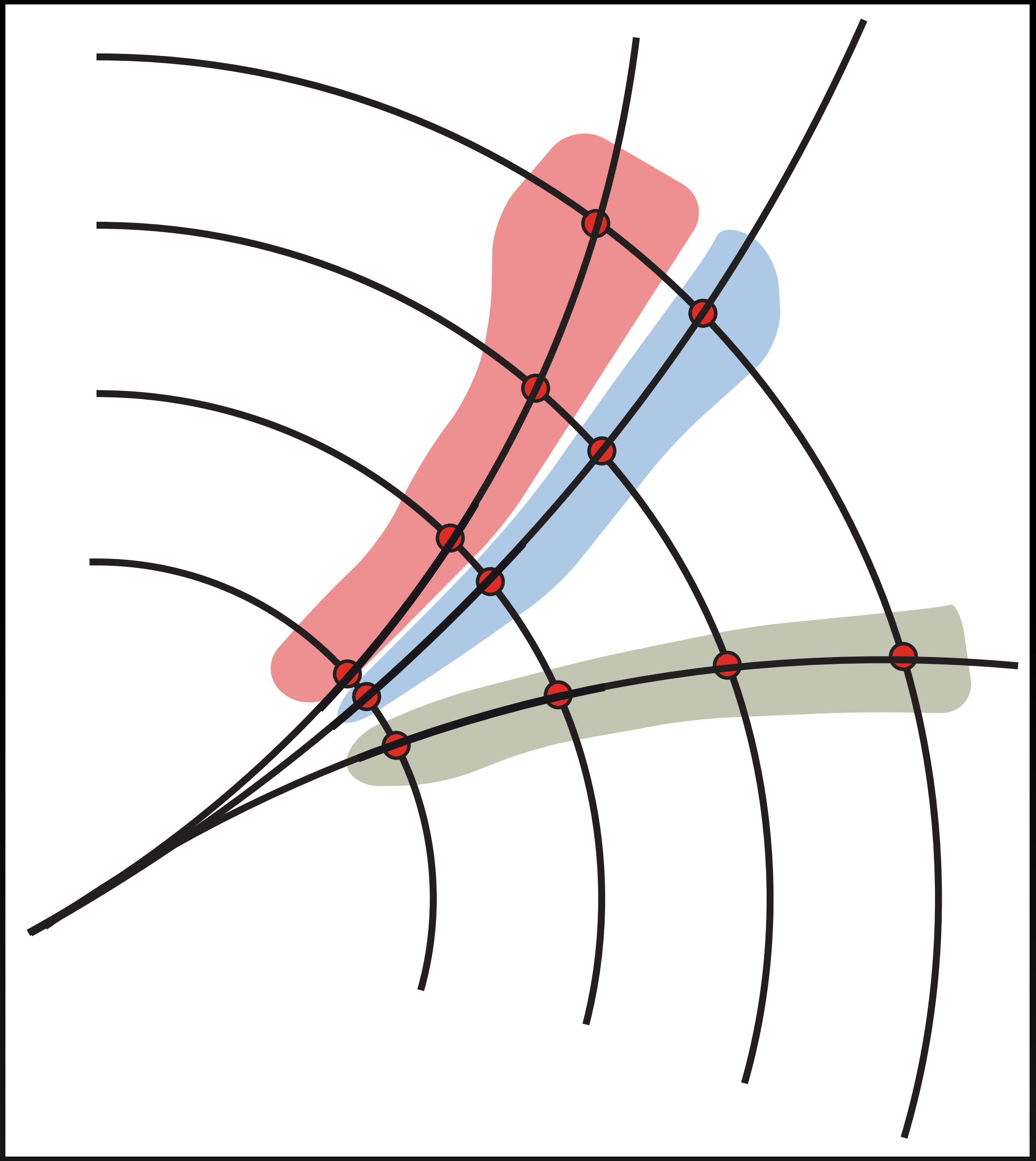
Tuning the Seeding Strategy

- the track finding algorithm
  - ➡ find **seed** from combination of 3 hits
    - search using hough transform
  - ➡ build **road** along the likely trajectory
  - ➡ run **combinatorial Kalman Filter** for a seed
    - full **exploration** of all possible candidates
    - update trajectory with hits at each layer
    - take material effects into account



Tuning the Seeding Strategy

- the track finding algorithm
  - ➡ find **seed** from combination of 3 hits
    - search using hough transform
  - ➡ build **road** along the likely trajectory
  - ➡ run **combinatorial Kalman Filter** for a seed
    - full **exploration** of all possible candidates
    - update trajectory with hits at each layer
    - take material effects into account



- iterative seeding approach (Run-1)
  - ➡ seeds are worked on in an ordered list
    - start with 3 Pixels, 2 Pixel+Strip, 3 Strips
  - ➡ bookkeeping layer:
    - hits from good candidates removed
    - build next seed ONLY from left over hits
  - ➡ sequential seed finding to avoid combinatorial explosion
    - unlike in the animation, tracks are found for one-after-the-other
    - hence, the ordering matters !!! (especially sorting in p_T bins)

Tuning the Seeding Strategy

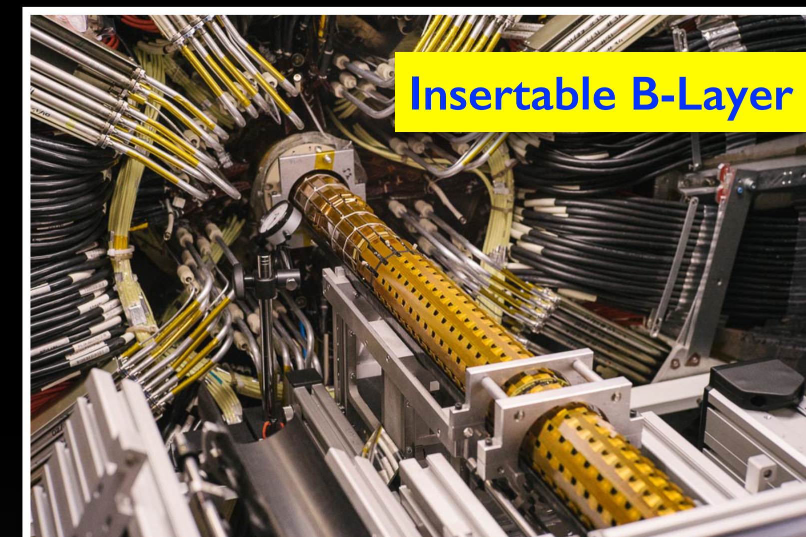
- optimal seeding strategy depends on level of pileup
  - ➡ efficiency of a seed to give a good track candidate:

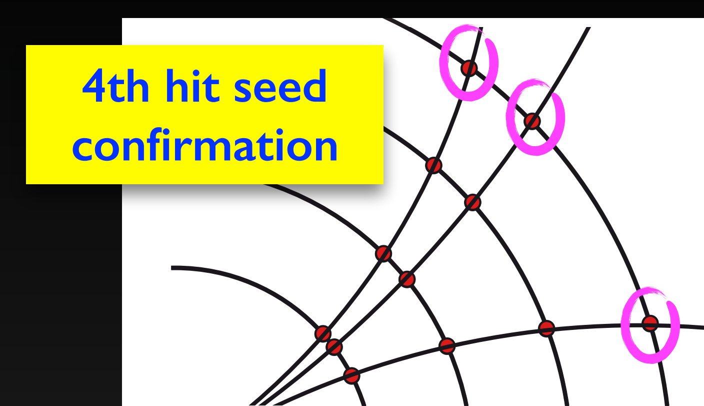
pileup	PPP	PPS	PSS	SSS
0	57%	26%	29%	66%
40	17%	6%	5%	35%

  - hence start with SSS at 40 pileup !
  - ➡ further increase seed efficiency using 4th hit

pileup	PPP+1	PPS+1	PSS+1	SSS+1
0	79%	53%	52%	86%
40	39%	8%	16%	70%

  - takes benefit from new Insertable B-Layer (IBL)





- final Run-2 seeding strategy
  - ➡ start with SSS+1
  - ➡ z(vertex) scan with found candidates
    - restrict seeding to z(first vertex) until z(last vertex)
  - ➡ continue with PPP+1, PPS+1, PSS+1

40 pileup @ 25 nsec

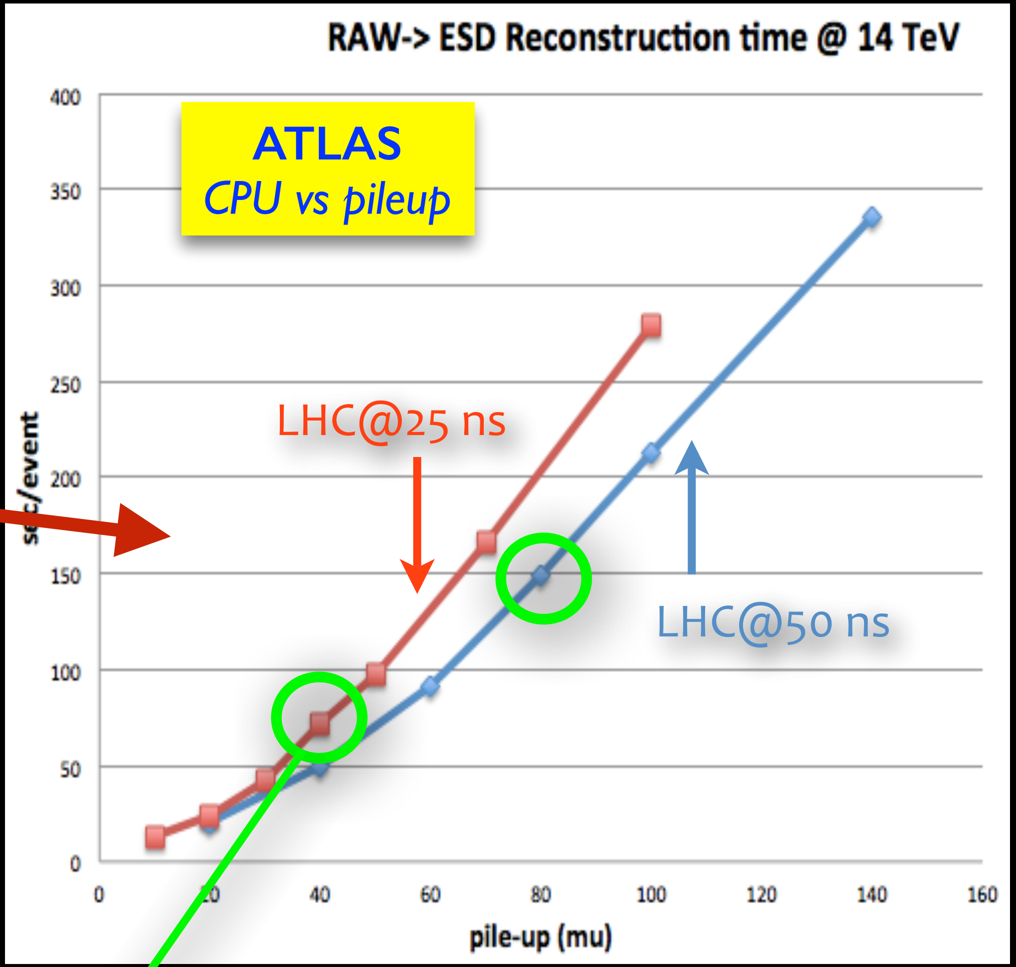
seeding	efficiency	CPU
"Run-1"	94.0%	9.5 sec
"Run-2"	94.2%	4.7 sec

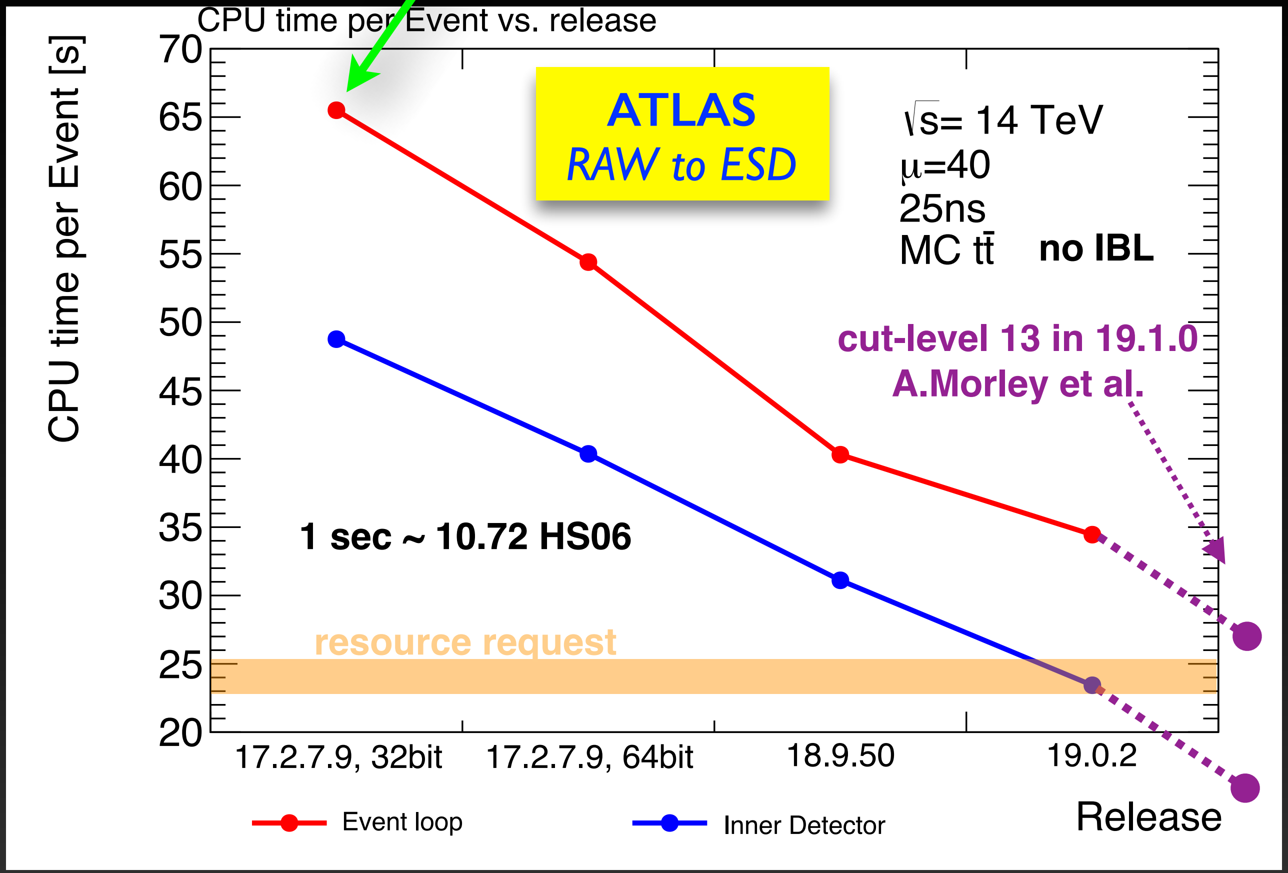
Igor Gavrilenko, CPU on local machine

Overall CPU Improvements

- tracking dominates in CPU vs pileup
  - ➡ Run-1 behaviour shown at the beginning
    - "combinatorial explosion" in hit combinations
- result of LS1 tracking upgrade project
  - ➡ touched more than 1000 packages !
  - ➡ technical and strategy improvements for 40 pileup
- on track for Tier-0 @ 1kHz:
  - ➡ CPU time on 14 TeV, ttbar, μ=40:
    - 17.2.7.9-32bit is the references (Tier-0)
    - 19.0.2 fully optimised for DC-14 / 8 TeV
    - setup for DC-14 / 13 TeV @ 40 pileup will be in 19.1.0

➡ 250 HS06/event within reach (CPU budget for 1 kHz @ Tier-0)

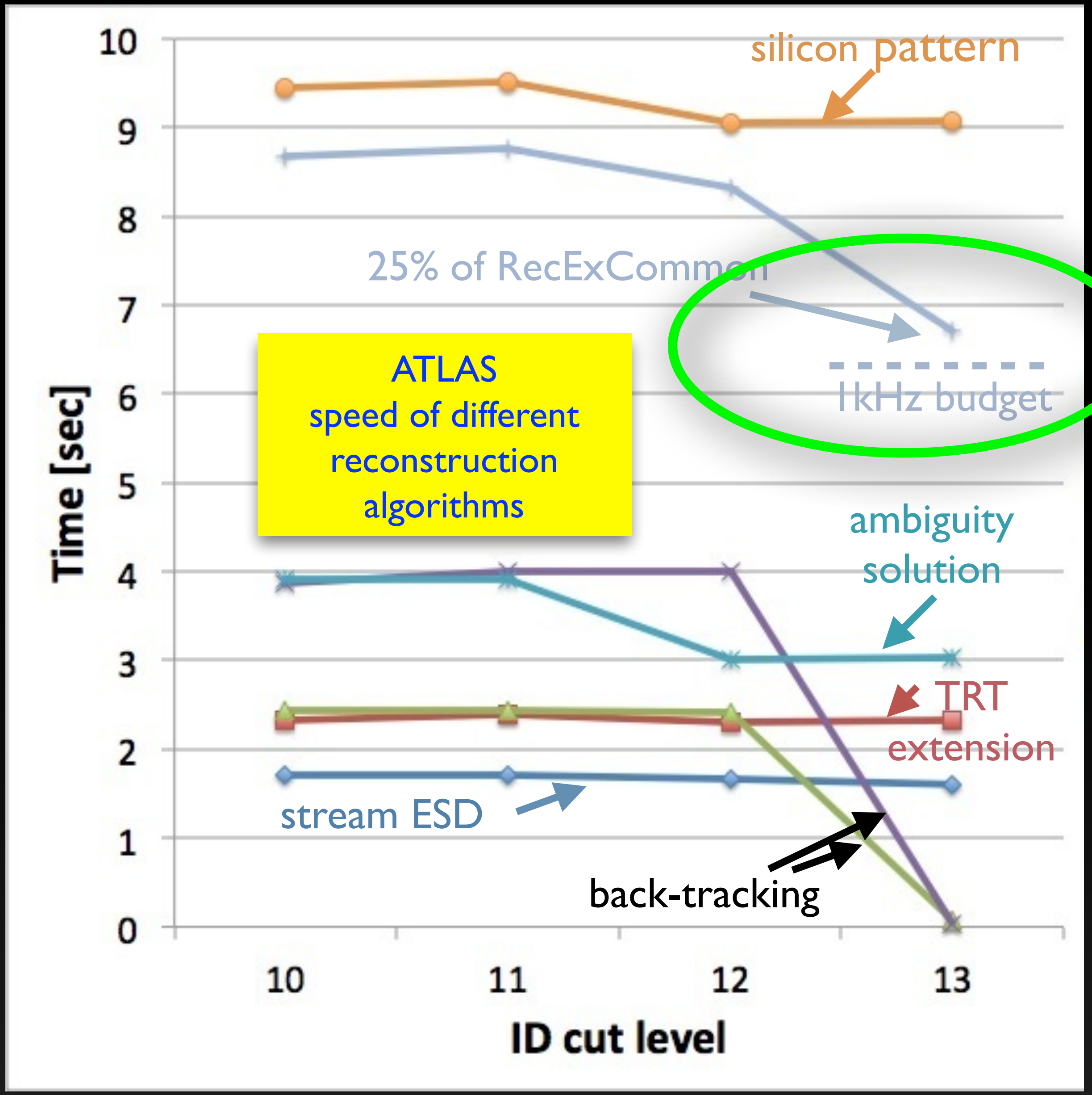




Rocco Mandrysch

Further CPU Improvements for 13 TeV

- tracking tuning for 13 TeV
  - ➡ release 19.0.X uses ID cut-level 10
    - includes Eigen, new seeding, ...
  - ➡ ID cut-level 13 for release 19.1.X
    - η-dependent TRT cuts
    - tuned silicon tracking cuts
    - back-tracking in EM RoIs (output tailored for e/gamma)
  - ➡ physics performance at μ=40 ?
    - better purity for primary tracks
    - e/gamma unchanged
  - ➡ RecExCommon with ID cut-level 13
    - <270 HS06 on 2012 high-μ run



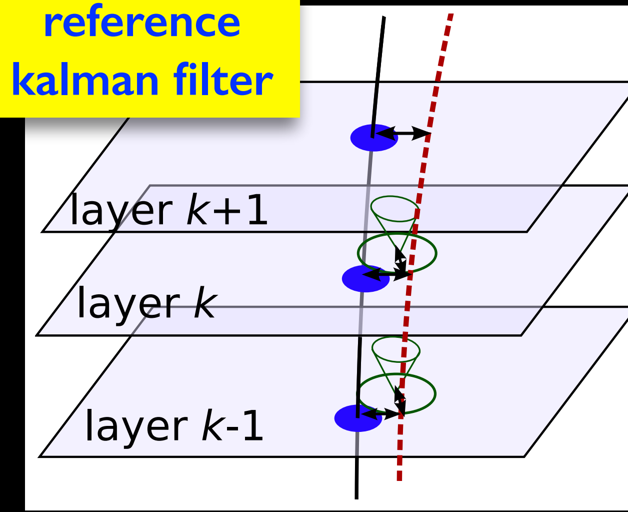
Anthony Morley et al., 2012 high-μ run

What is coming next ?

Further optimise current Tracking

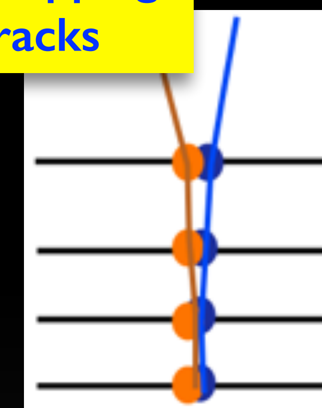
- algorithmic improvements being worked on
  - ➡ use only curvilinear frame inside extrapolator
    - saves local/global transformations
  - ➡ cache track extrapolation to calorimeter
    - extensively in combined reconstruction
  - ➡ faster track fit based on reference Kalman filter
    - linearise track fit w.r.t. reference trajectory (1 extrapolation)

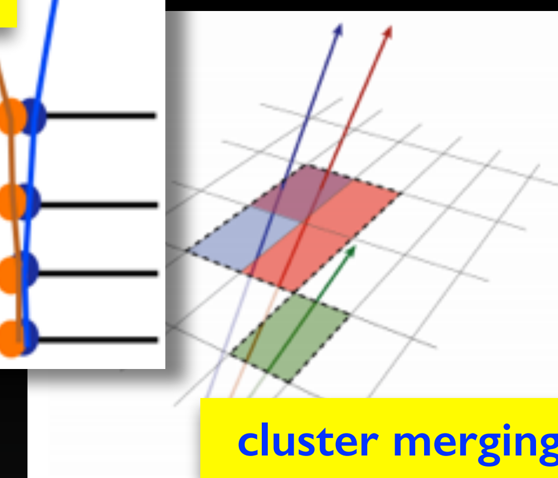
reference kalman filter



S.Fleischmann et al.

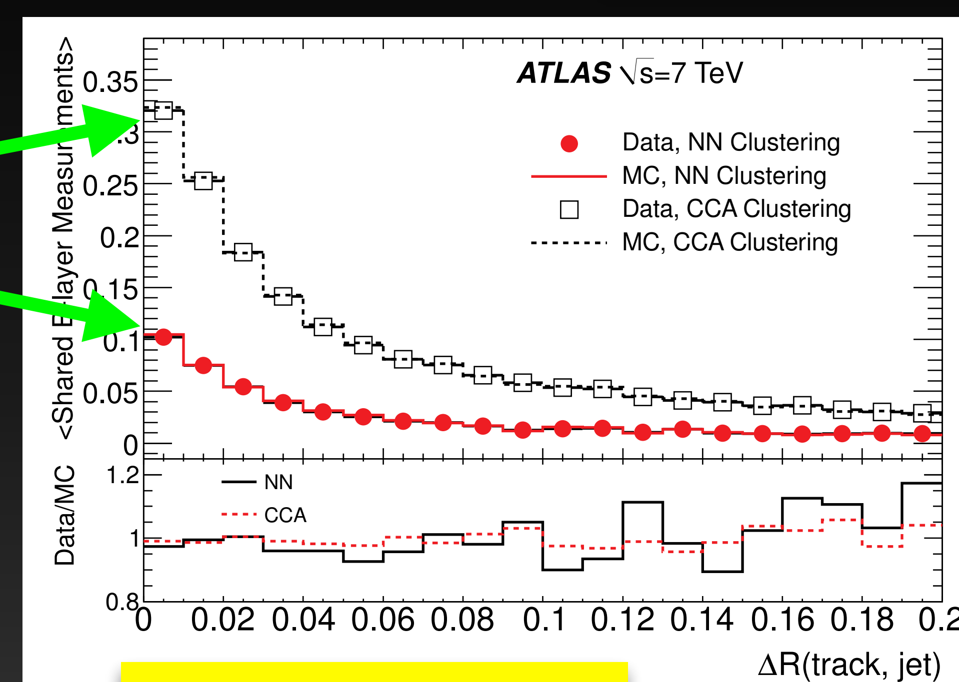
overlapping tracks





cluster merging

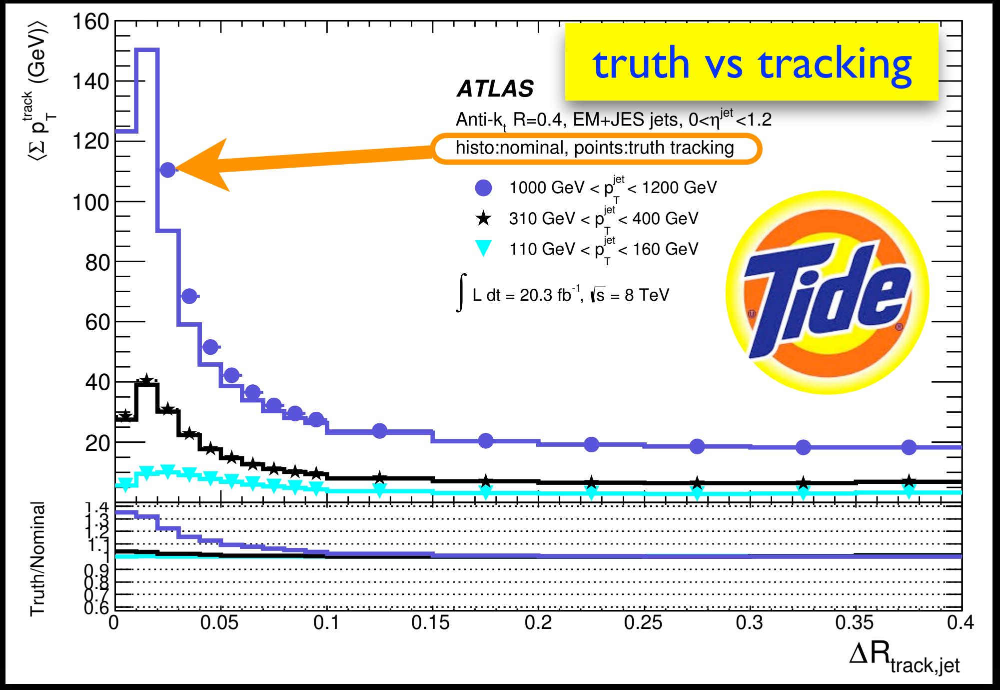
- explore ideas for tracking in jets
  - ➡ hit density in jet cores lead to cluster merging
    - reason for Neural Network (NN) cluster splitting
  - ➡ pattern usually finds track candidates
    - large number of shared hits still remain
  - ➡ task of ambiguity processing is to reject fakes
    - tracks with many shared hits looks like a fake
  - ➡ room for improvements ?



shared hits in jet cores with and without NN

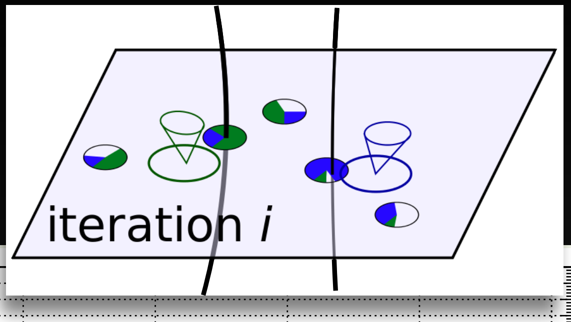
Tracking in dense Environments (TIDE)

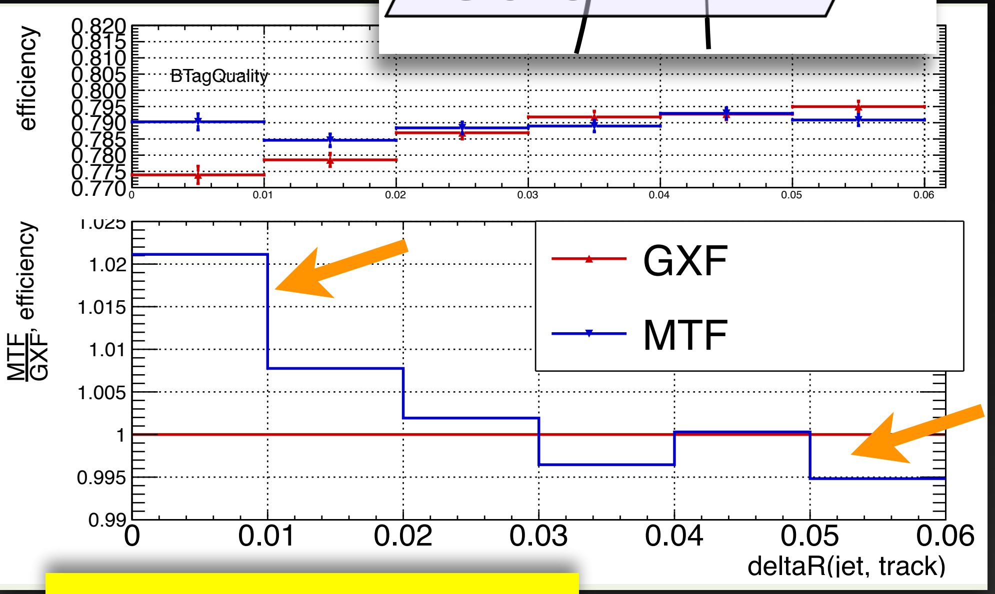
- try to improve in high-p_T jet RoI
  - ➡ TIDE working group
    - more elaborate ambiguity processing to recover tracks
    - especially relevant for high-p_T
  - ➡ aim to improve as well tau reconstruction
    - tracking inefficiencies limit for identification and particle flow (3 prongs)
  - ➡ truth tracking shows there is potential



- several strategies
  - ➡ improve selection and NN cluster splitting
    - aim is to keep more of the tracks with currently many merged/shared clusters
  - ➡ alternative algorithm: Multi Track Fitter (MTF)
    - robust (adaptive) version of Kalman filter
    - variant to estimate N tracks simultaneously can be use to resolve ambiguities (?)

M.Neumann, S.Fleischmann

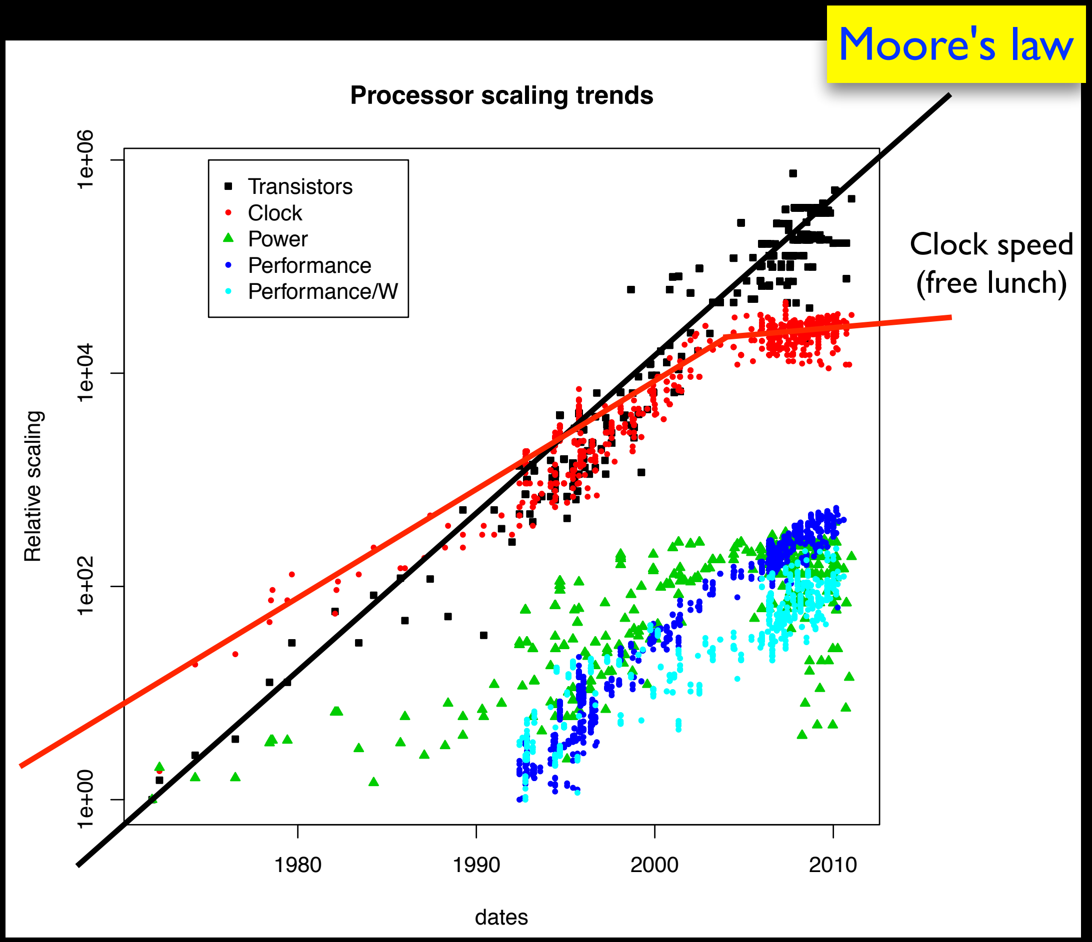




classical ambiguity vs MTF

Processor Technology

- Moore's law is still alive
  - ➡ number of transistors doubles every 2 years
  - ➡ lots of transistors looking for something to do:
    - vector registers
    - out of order execution
    - multiple cores
    - hyper threading
  - ➡ increase theoretical performance of processors
    - hard to achieve this performance with HEP applications !



- taking benefit from vector registers (SIMD)
  - ➡ Eigen and libimf used since release 19
    - internally vectorises computations, ~20% speedup seen
    - tracking code not yet optimised to exploit SIMD features
  - ➡ studies on hand-vectorising hot-spots like Runge-Kutta
    - needs experts to write SSE and AVX code
  - ➡ auto-vectorising using advanced compiler options
    - studies are ongoing, gains seen so far not too impressive

Data Locality

- Level-1 cache misses and data locality
  - ➡ ATLAS reconstruction has significant (2.2%) rate read/write cache misses
    - e.g., Runge-Kutta integration shows up high in cachegrind summary
  - ➡ studies show that this is very expensive
    - simple tests of sigmoid functions (for neural networks) with contiguous and random memory access:

Function	Contiguous	Random	Ratio
Logistical Fn	2400ms	9700ms	÷4
Fast sqrt Fn	560ms	7900ms	÷14
Ratio	x4.3	x1.2	

Improve because of SIMD

G.Stewart et al.

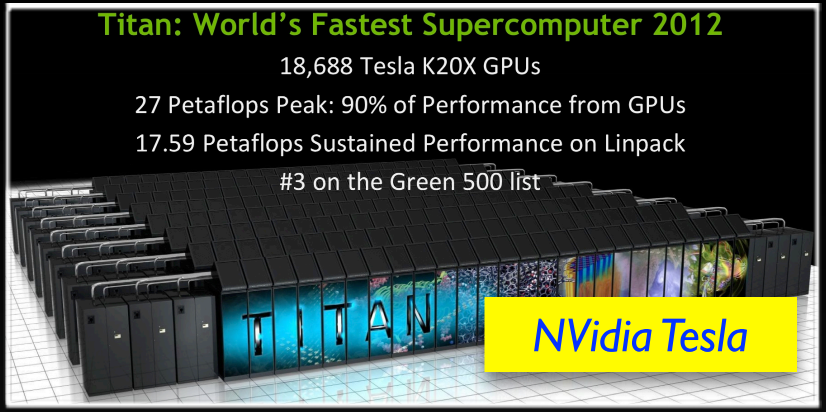
Losses because of lack of locality

- xAOD and data locality
  - ➡ separates API and data itself
    - interface class "electron"
    - data in "electronAuxStore"
  - ➡ AuxStore looks like "RootTuple"
    - data organised in a structure of vector<simple types>
  - ➡ idea is to enforce contiguous memory usage behind the scene
    - as well, data pools will reduce malloc overhead
    - requires to migrate remaining reconstruction EDM to xAOD format (clusters, drift circles, space points, tracks)
    - will as well help data reformatting for massively parallel processing (GPUs)

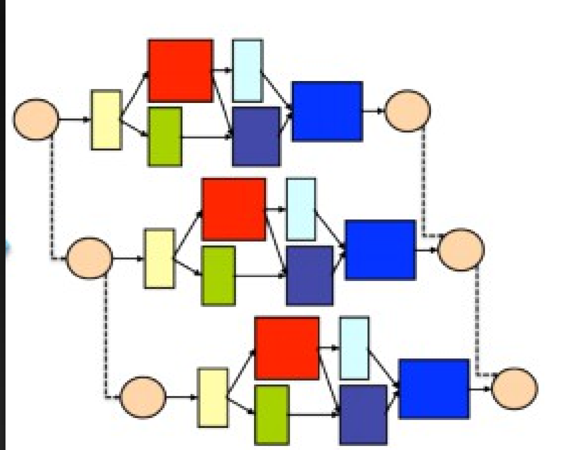
Tracking on
Many Core Processors

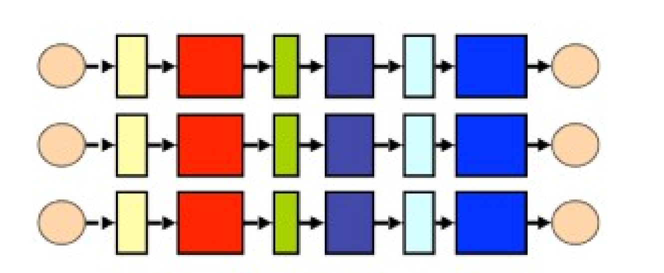
Multi-Processing and Multi-Threading

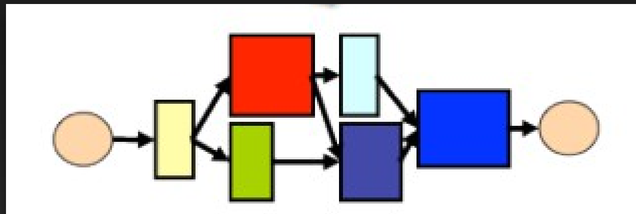
- many core processors, including GPGPUs
  - ➡ e.g. NVidia Tesla, Intel Phi
    - we see them in HPC applications
  - ➡ not so clear if and when they replace our GRID nodes



- lots of cores with little memory
  - ➡ need to parallelise application
    - same for ARM or ATOM processors with small memory
  - ➡ event-wise parallel processing (AthenaMP)
    - late process forking allows to share ~50% of memory
  - ➡ algorithm level multi-threading (Gaudi-Hive prototype)
    - concurrent processing supported by framework
    - tracking dominates, does not really "fit" Hive model (~85% of reconstruction are sequential algorithms)
  - ➡ ultimately, need multi-threading within algorithmic code


Parallelism at all levels


Simultaneous Multiple events


Parallelism within an event

Massively parallel Tracking

- ATLAS/CMS tracking strategy is for early rejection
  - ➡ iterative: avoid combinatorial overhead as much as possible !
    - early rejection requires strategic candidate processing and hit removal
  - ➡ not a heavily parallel approach, it is a SEQUENTIAL approach !
    - good scaling with pileup (factor 6-8 for 4 times pileup) - still catastrophic
- implications for making it massively parallel ?
  - ➡ Armdahl's law at work:

$$\mathrm{Time}_{||} = \mathrm{Para} / N + \mathrm{Seq}$$

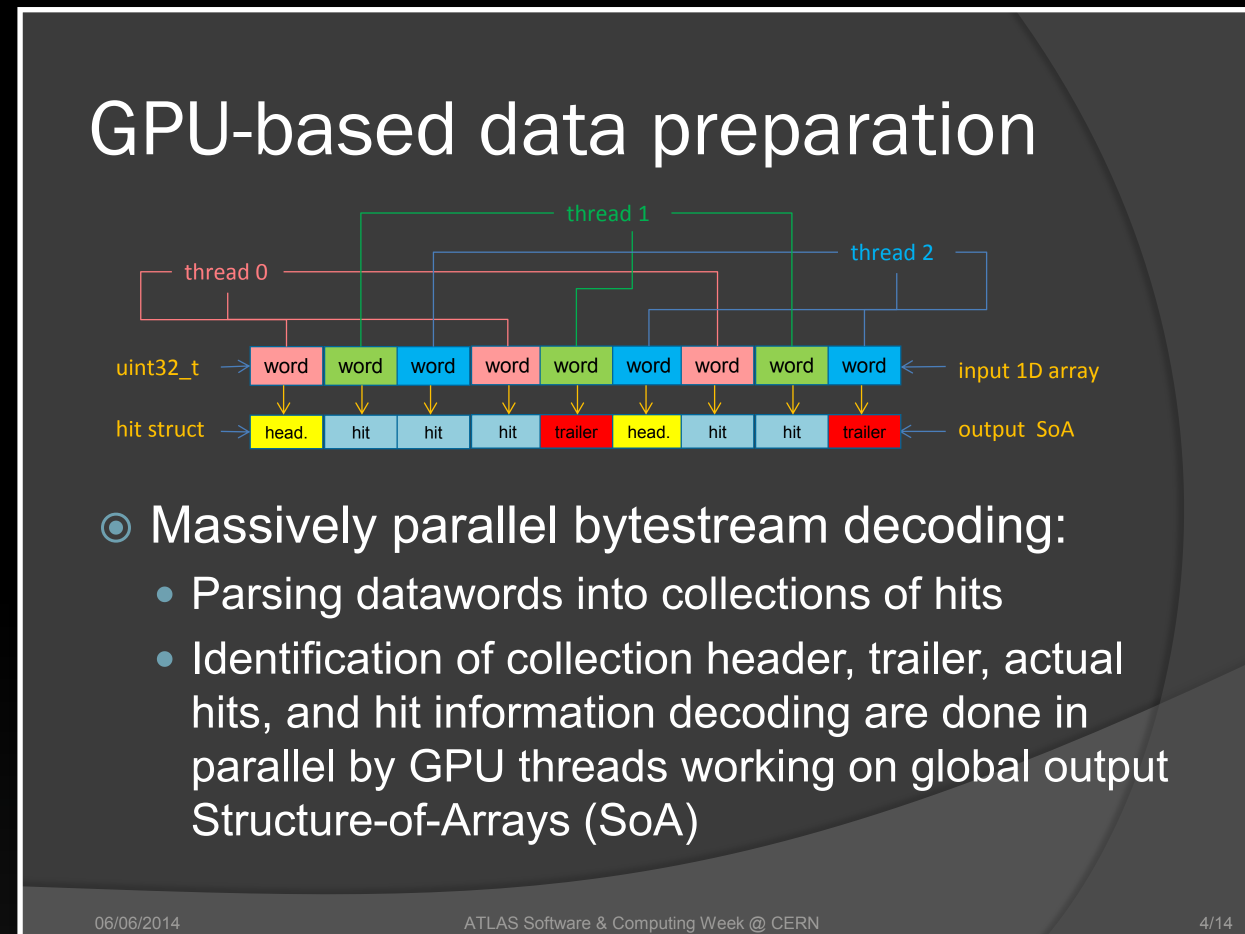
    - current strategy: small parallel part Par, while it is heavy on sequential Seq
  - ➡ hence: if we want to gain by a large N threads, we need to reduce Seq
    - compromise on early rejection, which means more combinatorial overhead
    - as a result, we will spend more CPU if we go parallel
  - ➡ makes sense if we use additional processing power that otherwise would not be usable (many core processors) or if latency is the main issue (trigger)
    - need to invest into R&D for novel parallel tracking strategies that reduce combinatorial overhead

Tracking on GPUs

- active field of development across experiments
  - ➡ see series of GSI Tracking Workshops (link to workshop)
    - collaboration between ALICE and FAIR on GPU tracking
    - ALICE already using GPU aided tracking in their trigger (PbPb)
  - ➡ within ATLAS several prototyping activities
    - Level-2 GPU tracking (RAL)
    - offline tracking studies (Mainz, Wuppertal, ...)
  - ➡ as well, studies on GPU integration
    - client/servicer architecture APE (RAL)
    - using dOpenCL communication layer (Münster, Wuppertal)
- within ATLAS Level-2 GPU tracking is most advanced
  - ➡ 2 years for complete re-write of Level-2 code for GPUs (D.Emeliyanov)
    - compact representations of geometry, b-field, cabling suitable for GPU
    - lightweight data structures for the on-GPU data model with conversion from/to Athena EDM
    - complete code re-factoring to get rid of "spaghetti" design, multiple loops, recursive calls

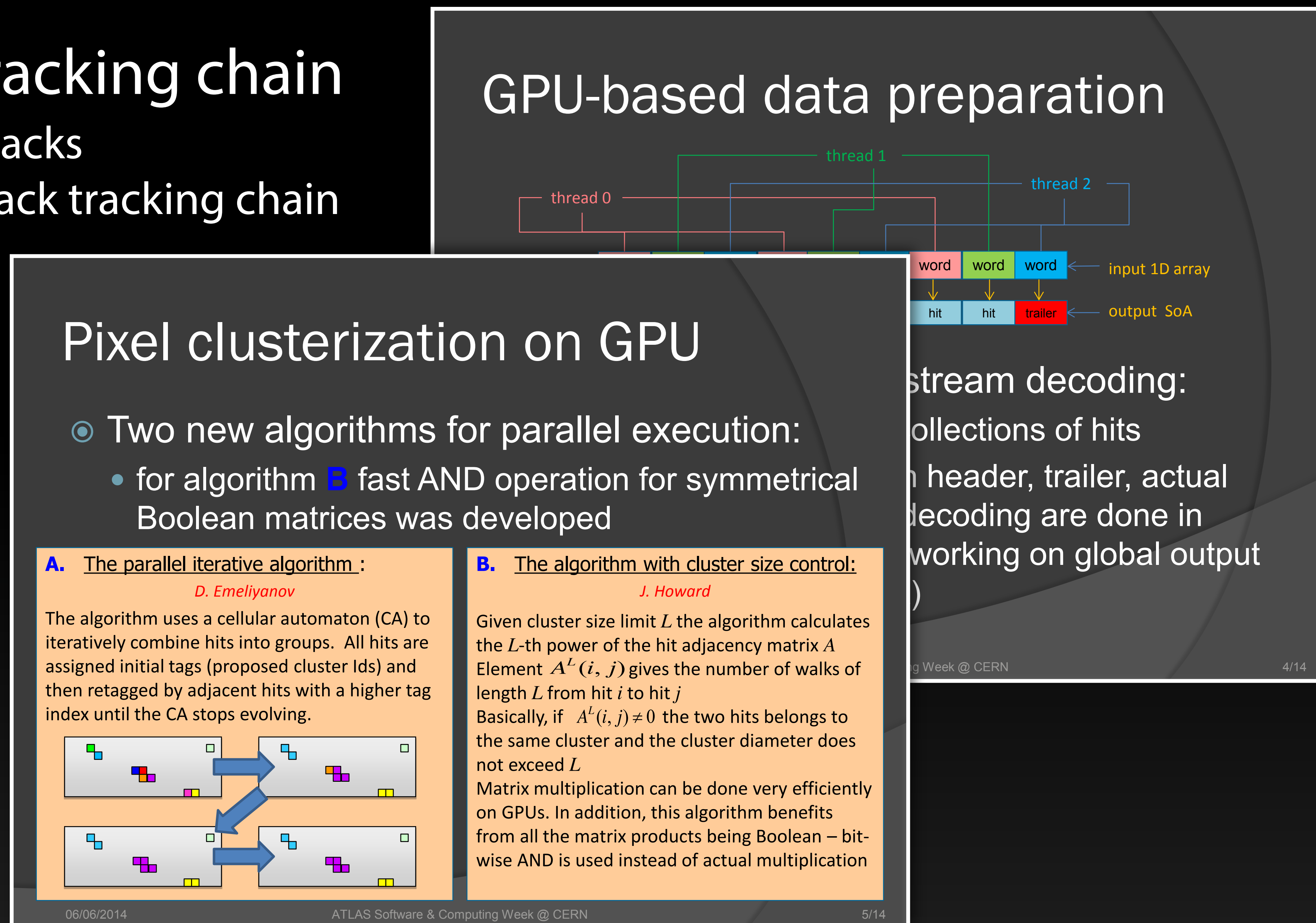
Level-2 GPU Tracking Prototype

- complete tracking chain
  - ➡ from raw to tracks
  - ➡ similar to SiTrack tracking chain



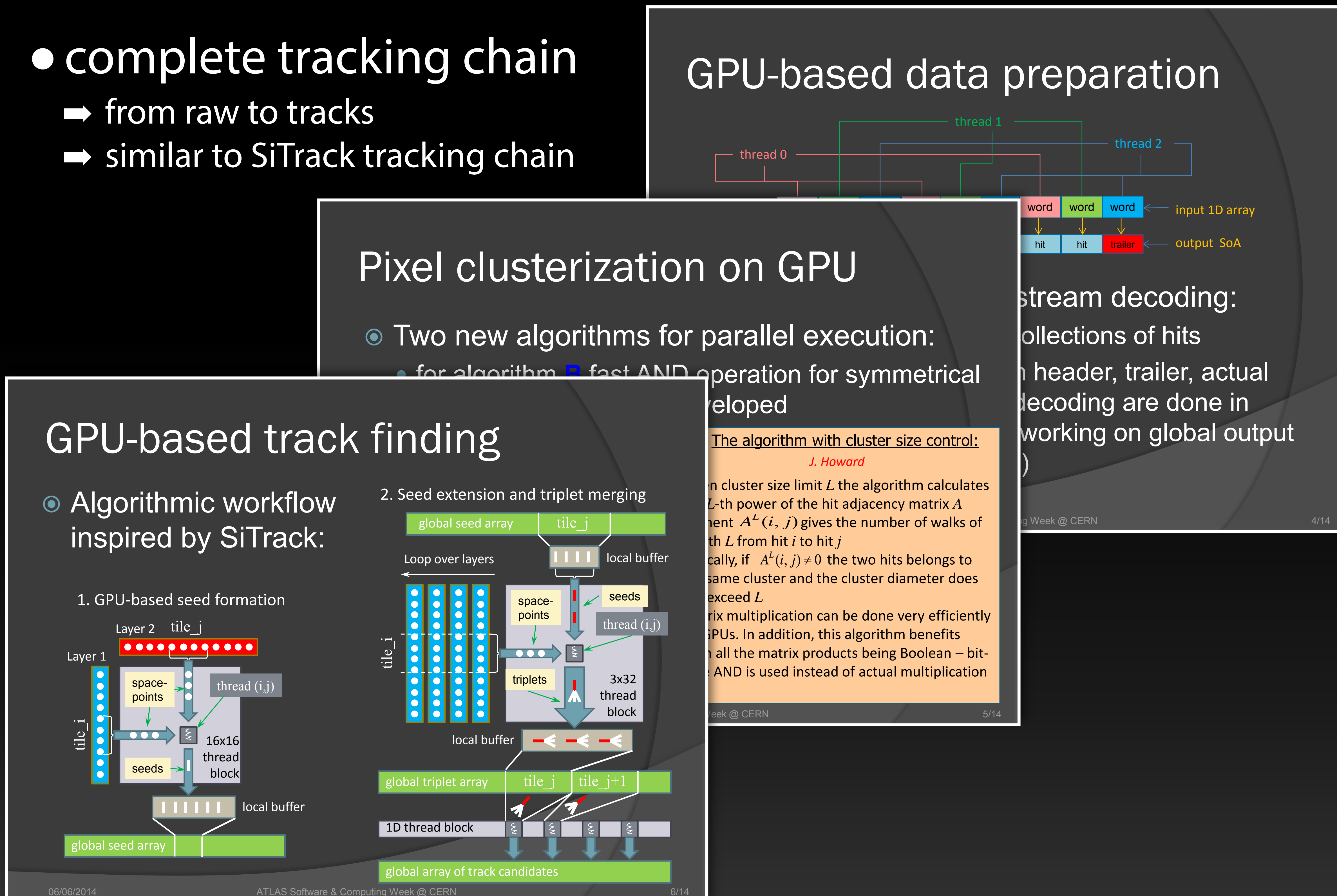
Level-2 GPU Tracking Prototype

- complete tracking chain
  - ➡ from raw to tracks
  - ➡ similar to SiTrack tracking chain



Level-2 GPU Tracking Prototype

- complete tracking chain
  - ➡ from raw to tracks
  - ➡ similar to SiTrack tracking chain

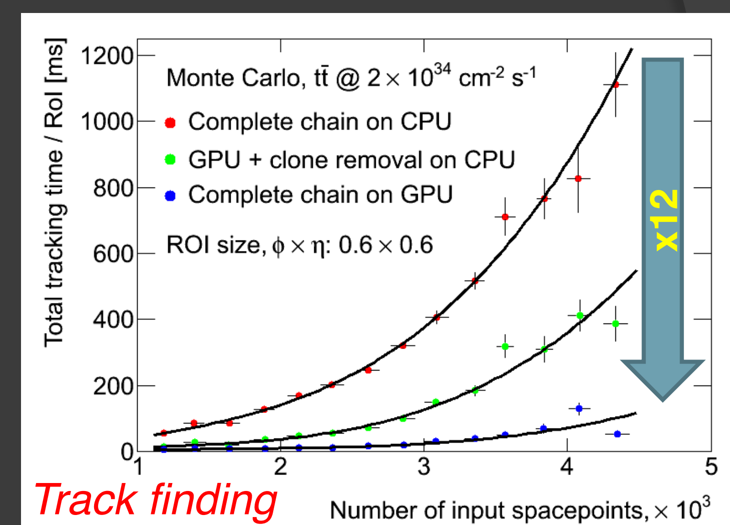


Level-2 GPU Tracking Prototype

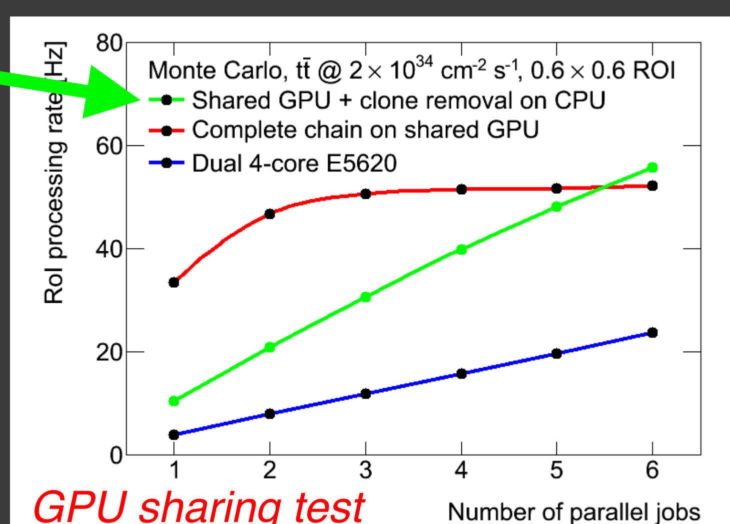
Summary of the results

- GPU-based code vs. 32-bit Athena (17.1.0)

RoI type	Data prep. speed-up
Tau 0.6x0.6	9
B-phys, 1.5x1.5	12
FullScan	26



sequential part on CPU



- x12 speed-up was obtained for the full LVL2 ID tracking chain on large RoIs
- "Client-server" architecture for GPU sharing seems to be feasible

06/06/2014 ATLAS Software & Computing Week @ CERN 7/14

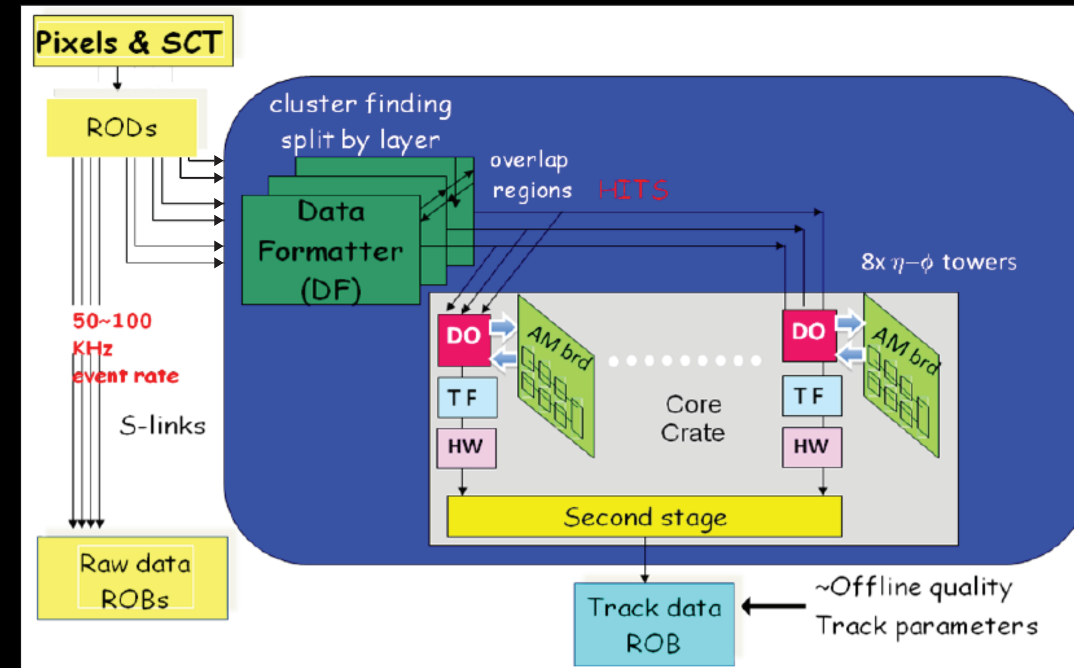
➡ significant speedup compared to running same chain on CPU
➡ CUDA vs openCL, development and maintenance cost ?

Tracking and Detector Upgrades

Hardware Solutions to Tracking ?

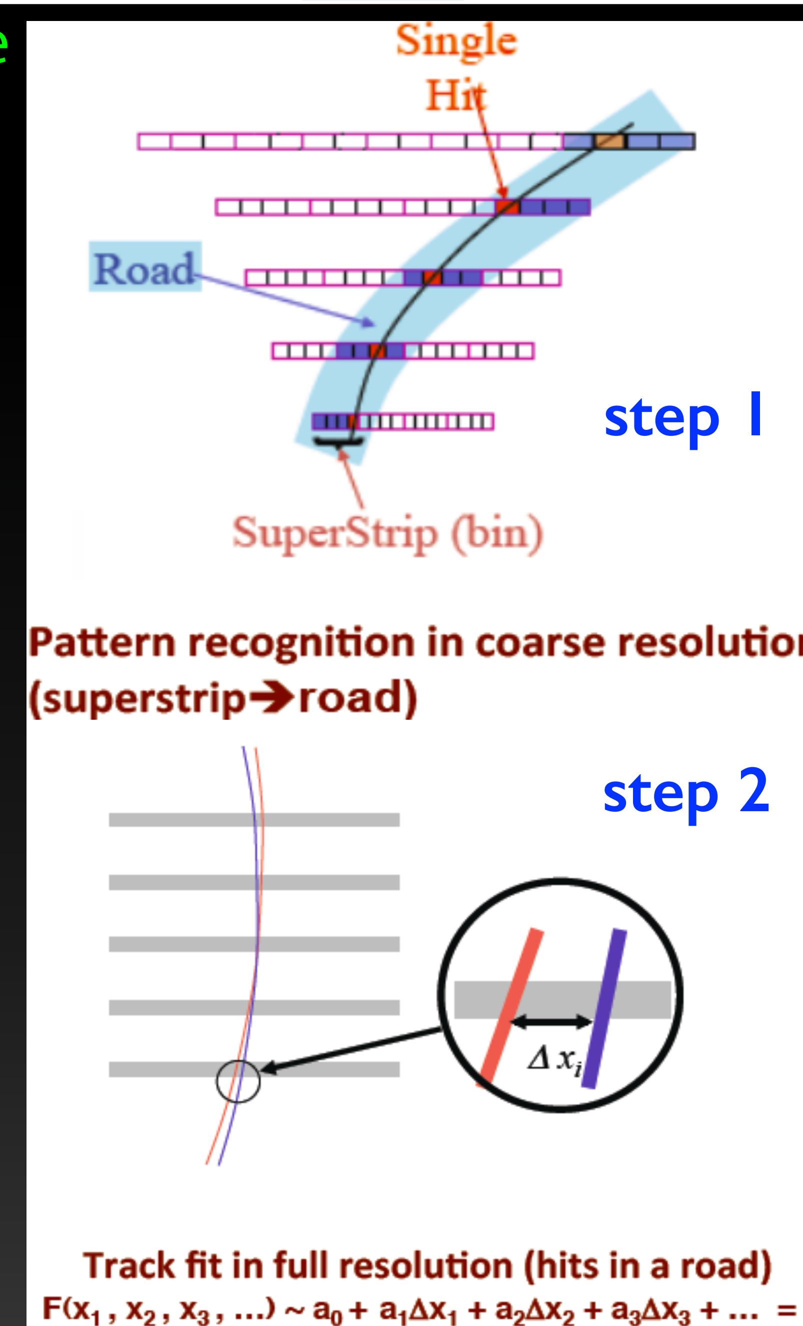
- using hardware tracking (FTK) ?
  - ➡ once installed, FTK will process every Level-1 trigger on data
    - will replace parts of Level-2 tracking
    - may be used to seed Event Filter tracking (Level-2 seeding being studied)
    - but: physics performance not matching offline
  - ➡ FTK simulation is SLOW on CPUs, factors >> 10 compared to offline
    - will not be able to process every MC event with FTK simulation
    - but: we could use time between fills to process MC in FTK at Point-1
    - or go crazy: build a 2nd FTK for processing MC
  - ➡ I would conclude: FTK not a drop-in solution to offline tracking problem
- optimising hardware for tracking ?
  - ➡ definitely !
    - ITK layout was optimised having robustness and tracking in mind
    - we could probably still do better, technology (CMOS) for all Pixels/Strixels
  - ➡ but: CMS is backing off from their L1 tracking dominated design
    - too restrictive in terms of physics performance, need to keep balance !

The Fast Tracker (FTK)



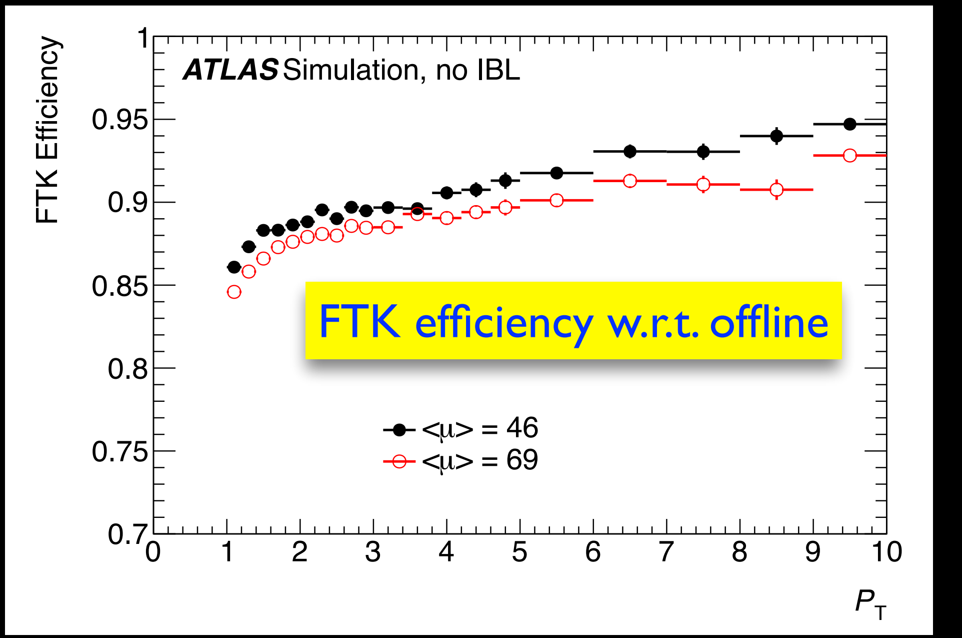
- current ATLAS trigger chain
  - ➡ Level-1: hardware based (~50 *kHz*)
  - ➡ Level-2: software based with RoI access to full granularity data (~5 *kHz*) tracking enters here
  - ➡ Event Filter: software trigger (~500 *Hz*)

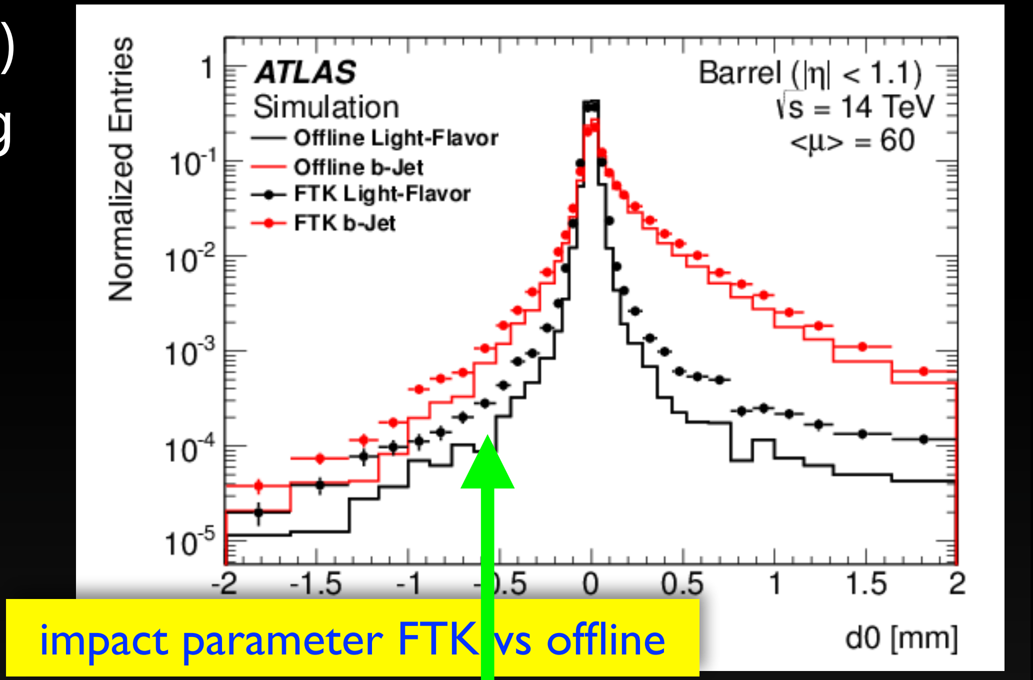
- FTK: hardware tracking (co-processor)
  - ➡ descendent of the CDF Silicon Vertex Trigger (SVT)
  - ➡ inputs from Pixel and SCT
    - data in parallel to normal read-out
  - ➡ two step reconstruction
    - associative memories for parallel pattern finding
    - linearised track "fit" implemented in FPGAs
  - ➡ provides track information to Level-2 in ~ 25 *μs*
    - slice installed for 2015, full coverage in 2016

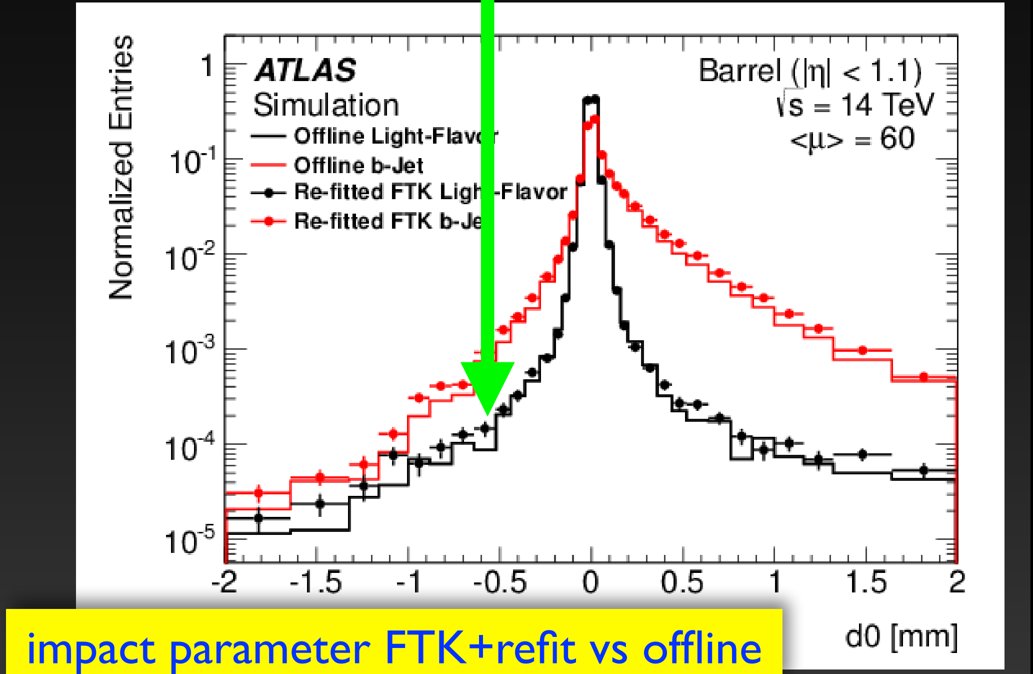


FTK Performance

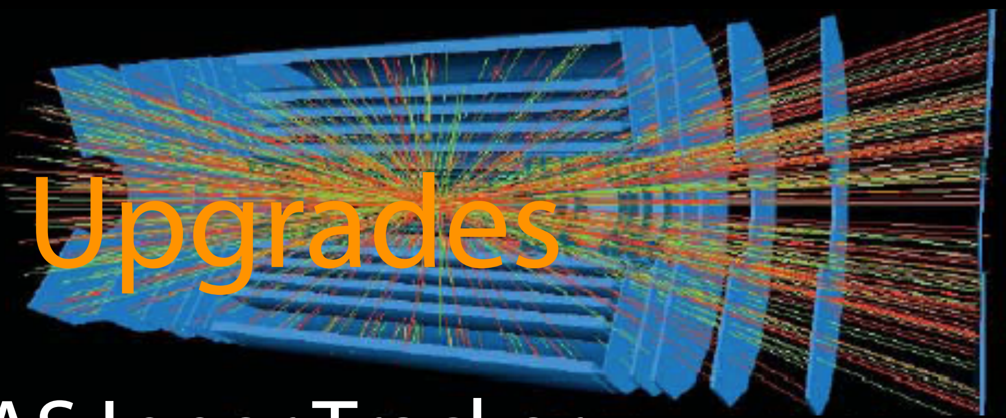
- effects and expected performance
  - ➡ track efficiency is 90-95% w.r.t. offline (loose match)
    - reduced detector granularity for track finding
    - size of candidate pattern banks is limited (20GB)
    - fast "hit worrier" vs offline ambiguity processing
  - ➡ track resolution (tails) limited by FPGA technique
    - track fit is linear estimator, not a real χ^2 track fit
    - not full resolution, no explicit material effects
- FTK still very useful for trigger
  - ➡ full scan at entry to Level-2
    - pileup corrections for jet and missing ET
    - particle flow like tau tagging (RoI as well ok ?)
    - fast track confirmation of Level-1 triggers
  - ➡ can recover offline like track resolutions
    - refit FTK tracks with Level-2 track fit
    - b-jet trigger, taus...





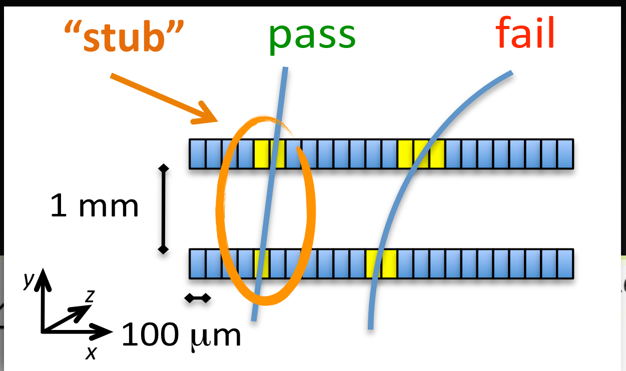


ATLAS and CMS Inner Tracker Upgrades

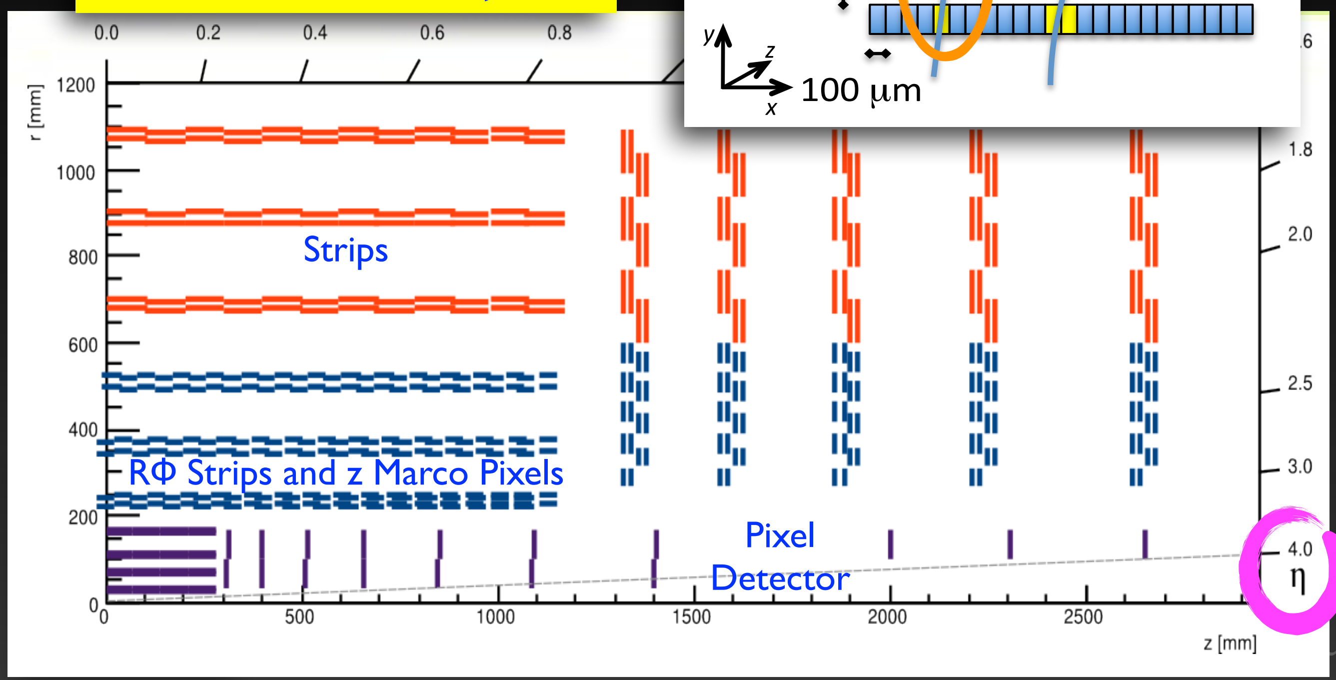


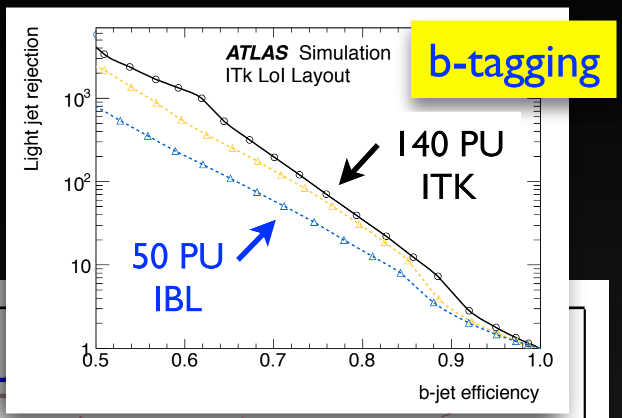
- CMS Inner Tracker
  - ➡ Strip tracker replacement
    - several layouts under consideration
    - short strips in $R\phi$, macro-pixels in z
  - ➡ Level-1 track trigger with high p_T stubs
    - correlate 2 sensors, threshold ~ *2 GeV*
    - pattern in associative memory, FPGA fit
  - ➡ Pixels: extend η coverage to 4 (!)

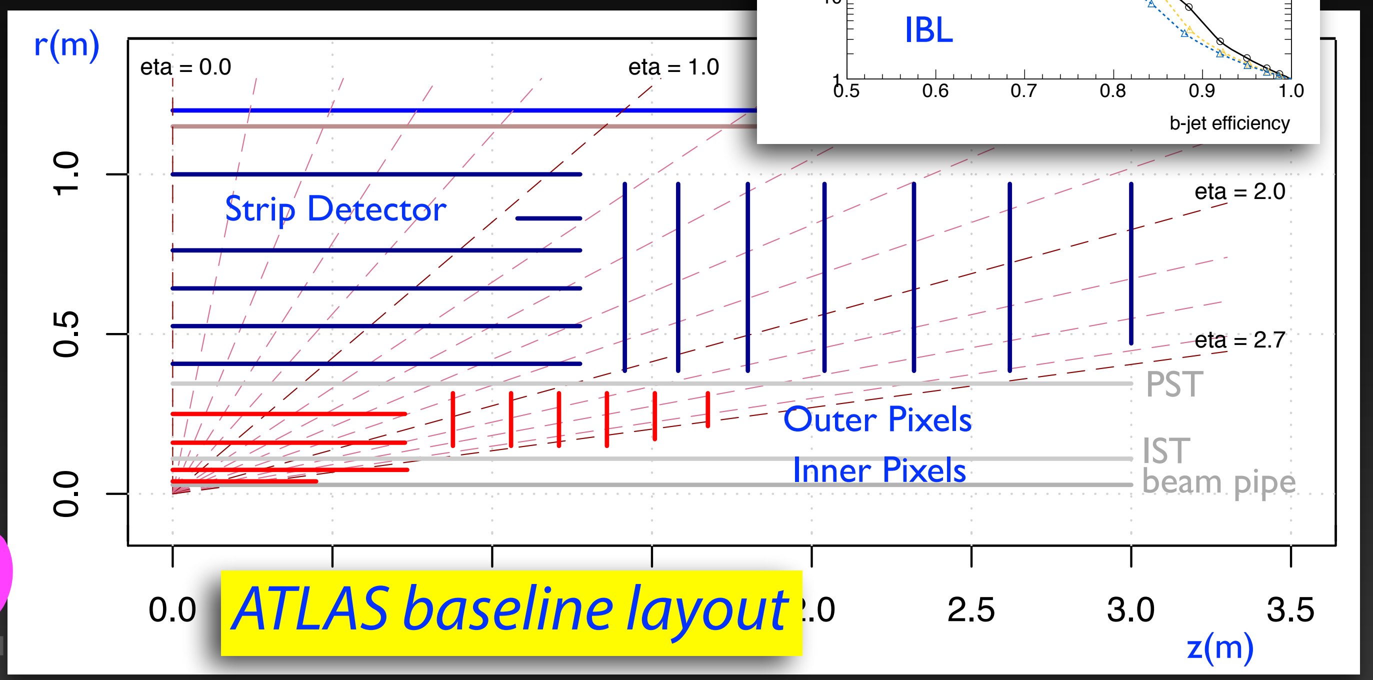
- ATLAS Inner Tracker
  - ➡ baseline: all silicon tracker, 14 hits
    - robust tracking @140 PU for η<2.5
  - ➡ Strip tracker with short strips + stereo
  - ➡ Pixels cover η<2.7 (Muons)
    - inner Pixels replaceable, reduced pitch
    - alternative layouts (“Alpine”, conical)
  - ➡ Level-1 track trigger seeded by Level-0
    - FTK inspired, reduced latency



CMS baseline layout

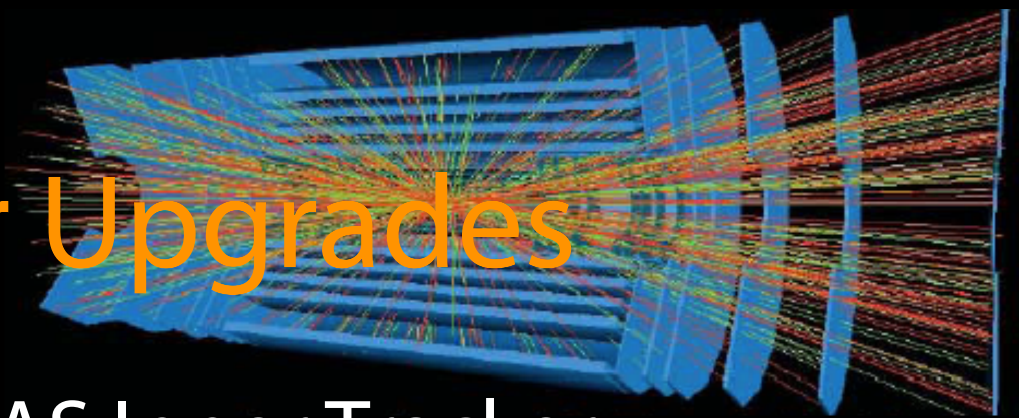






ATLAS baseline layout

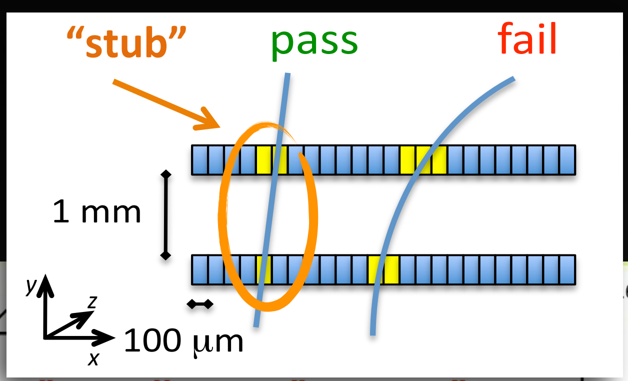
ATLAS and CMS Inner Tracker Upgrades

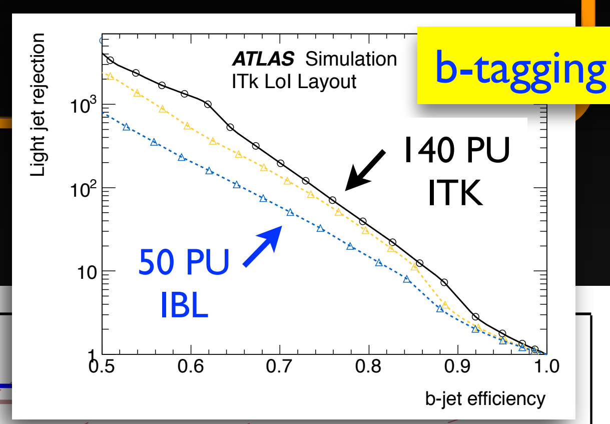


- CMS Inner Tracker
  - ➡ Strip tracker replacement
    - several layouts under consideration
    - short strips in $R\phi$, macro-pixels in z
  - ➡ Level-1 track trigger with high p_T stubs
    - correlate 2 sensors, threshold ~ *2 GeV*
    - pattern in associative memory, FPGA fit
  - ➡ Pixels: extend η coverage to 4 (!)

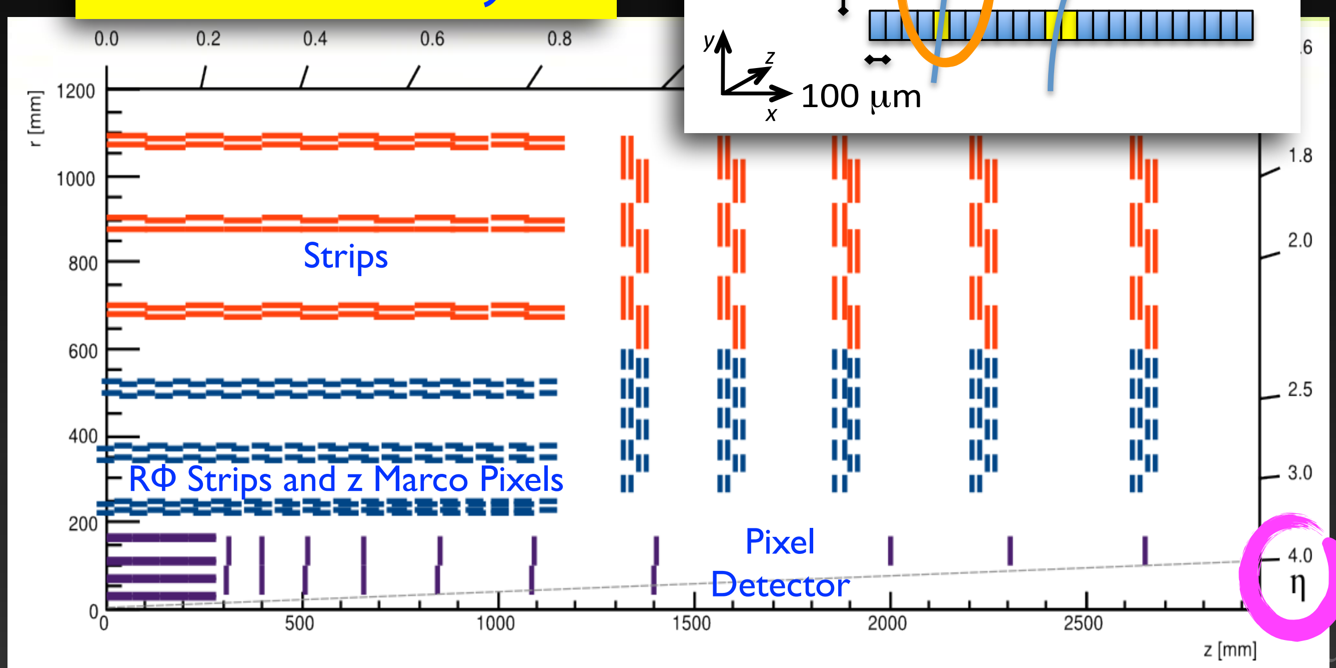
optimised for fast (HW) tracking

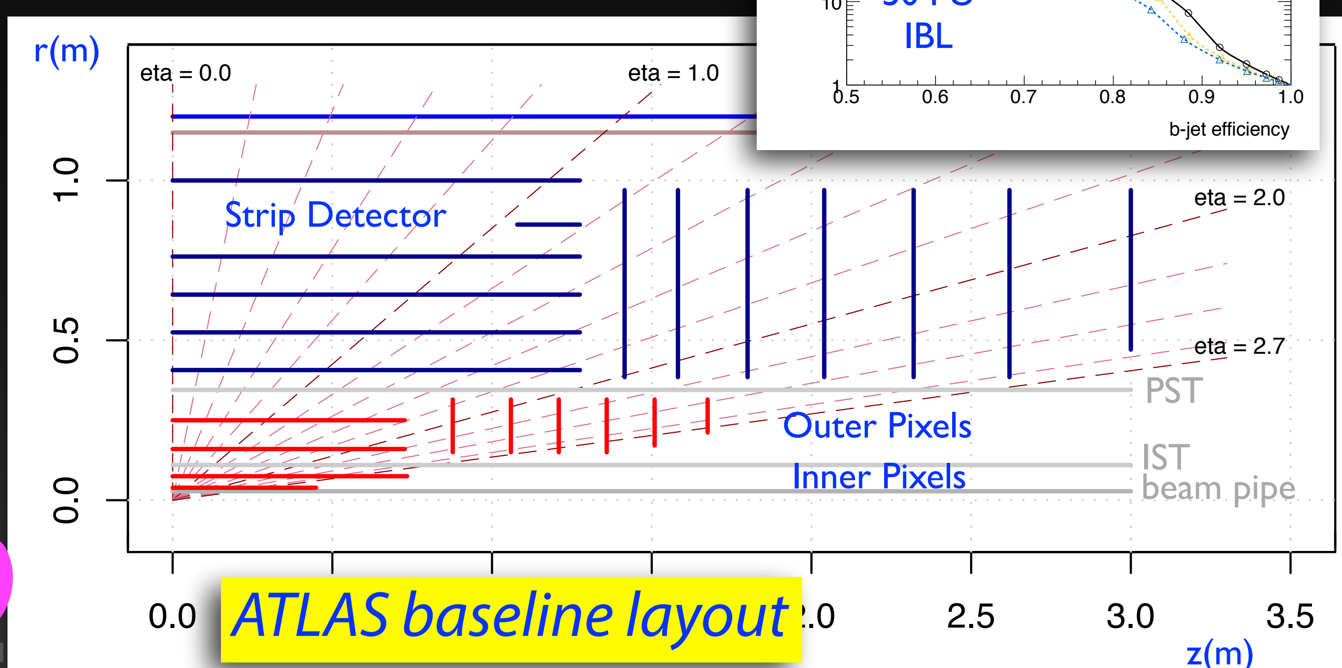
- ATLAS Inner Tracker
  - ➡ baseline: all silicon tracker, 14 hits
    - robust tracking @140 PU for η<2.5
  - ➡ Strip tracker with short strips + stereo
  - ➡ Pixels cover η<2.7 (Muons)
    - inner Pixels replaceable, reduced pitch
    - alternative layouts ("Alpine", conical)
  - ➡ Level-1 track trigger seeded by Level-0
    - FTK inspired, reduced latency





CMS baseline layout



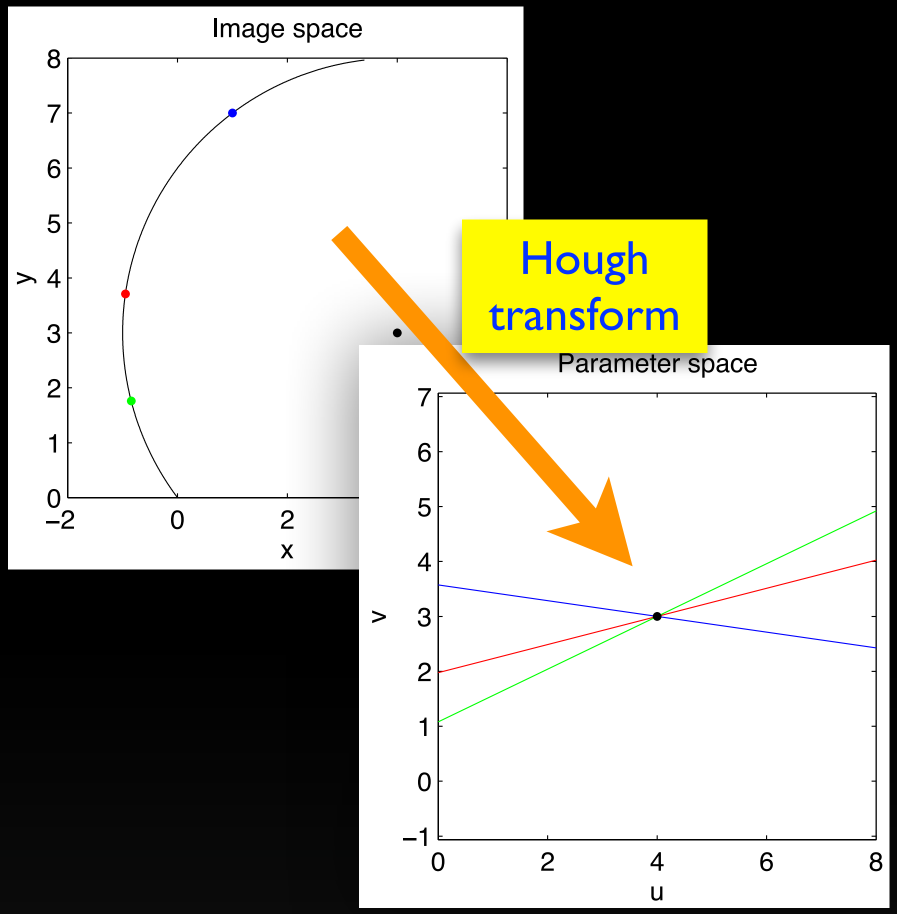


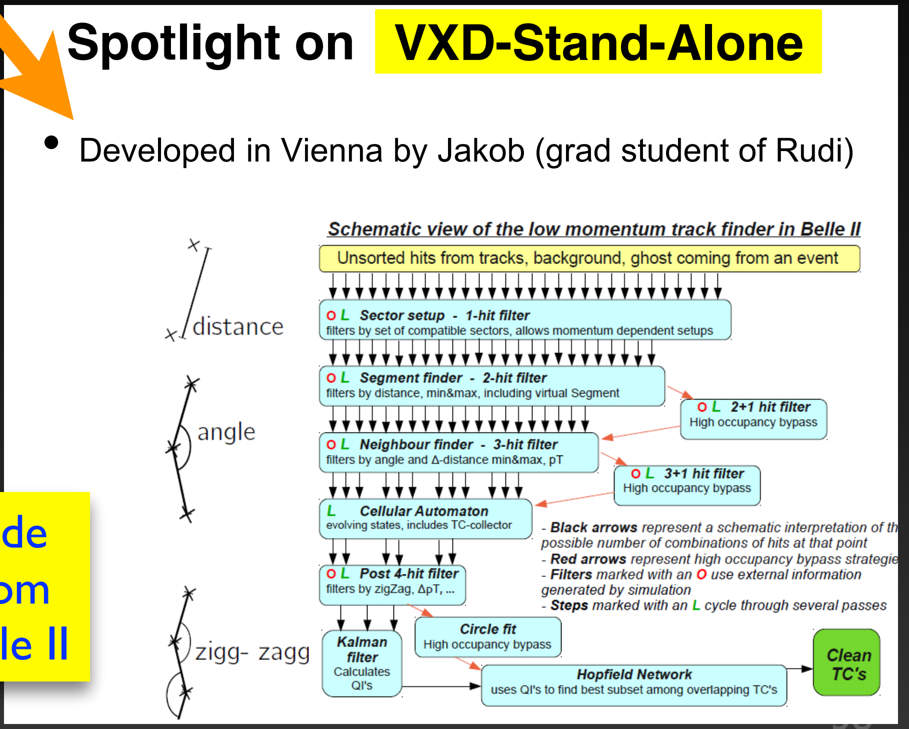
ATLAS baseline layout

New Ideas for Track Reconstruction ?

Alternative Tracking Algorithms

- examples for algorithms in literature
  - ➡ conformal transforms: e.g. Hough transforms
    - scale ~ linear with pileup, need memory
    - used in track seeding and TRT segment finding
    - no successful application for full Pixels+SCT yet
  - ➡ still transforms: V-trees
    - scale ~ linear with pileup
    - used in IDSCAN for Level-2 tracking
    - intrinsically pointing, needs primary vertex
  - ➡ cellular automaton
    - used by some experiments, example Belle II (not their default tracking code !)
    - idea is to evolve 3 hit combinations into tracks
    - it's a combinatorial algorithm that could be parallelised
    - Belle II example uses things like "high occupancy bypasses" in their algorithm flow ?





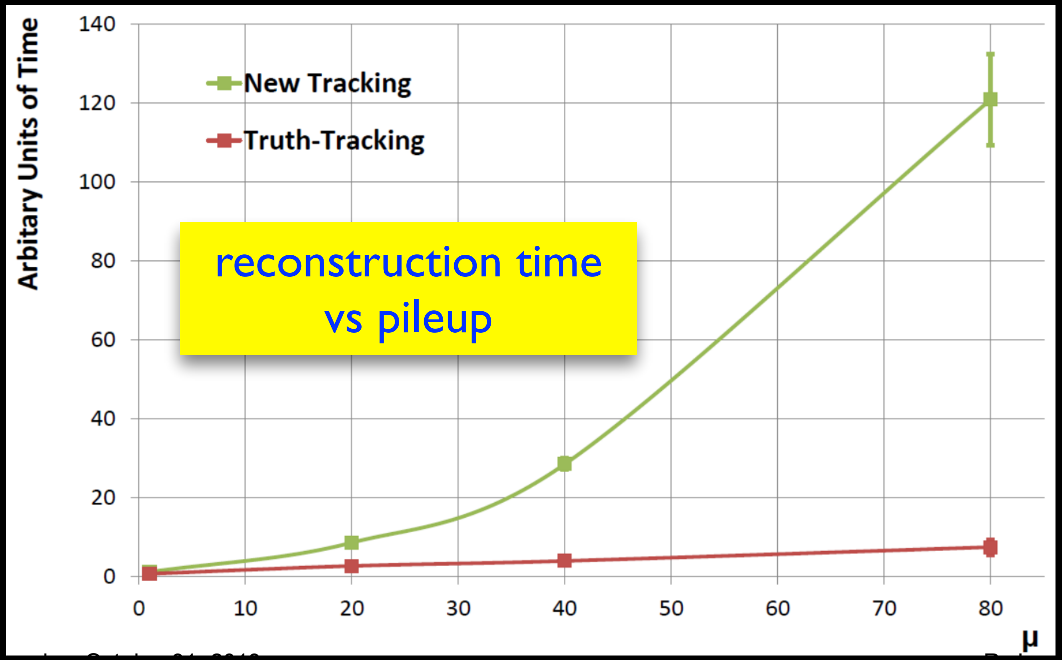
slide from Belle II

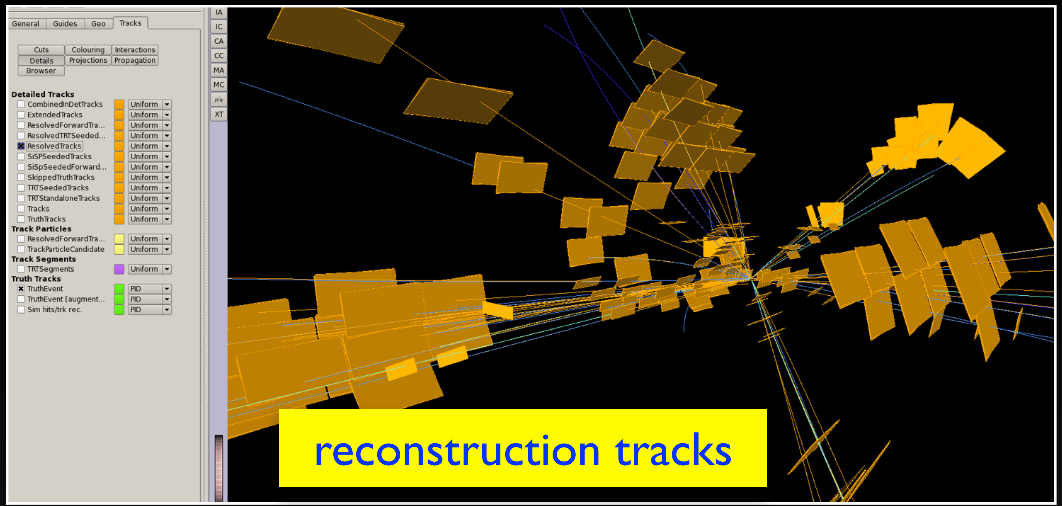
- we probably need new ideas !

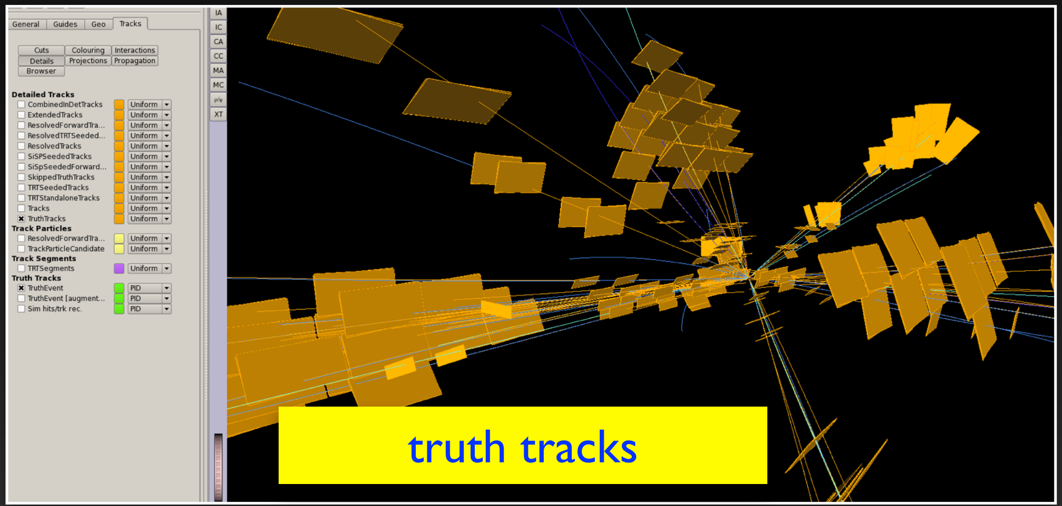
CERN

Truth Tracking from MC

- for very fast (ISF) simulation options
  - ➡ MC truth based hit filter to find tracks
  - ➡ replace pattern recognition in tracker
    - otherwise limiting CPU driver
- good results achieved
  - ➡ real pattern is very efficient and very pure
    - modeling of hit association mostly ok
  - ➡ models main source of inefficiencies well
    - this is hadronic interactions in material
  - ➡ uses full fit, so resolution come out right
  - ➡ and it is fast (trivial) !
- still, corrections are needed
  - ➡ especially double track resolution
    - affects jet cores, taus, maybe 140 pileup (?)
  - ➡ corrections are topology dependent

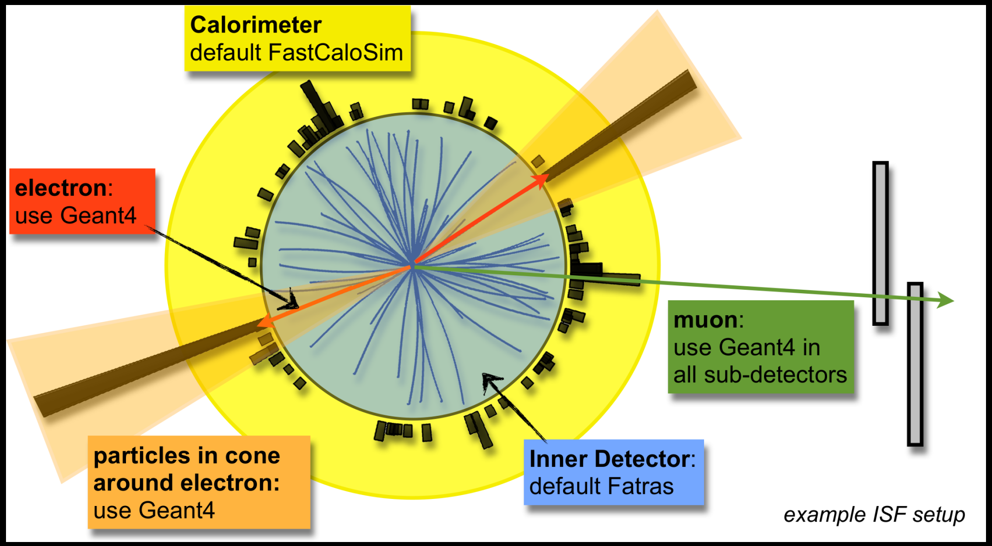






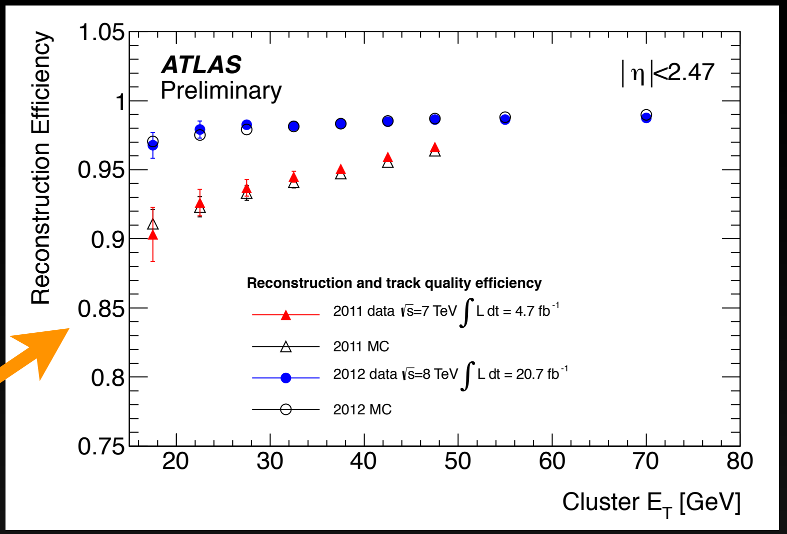
The ISF Idea for Tracking ?

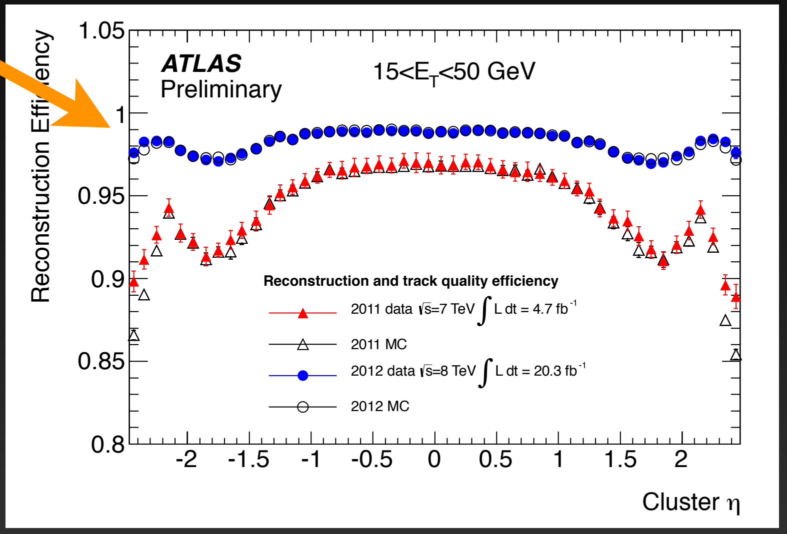
A.Salzburger



E.Ritsch, A.Salzburger

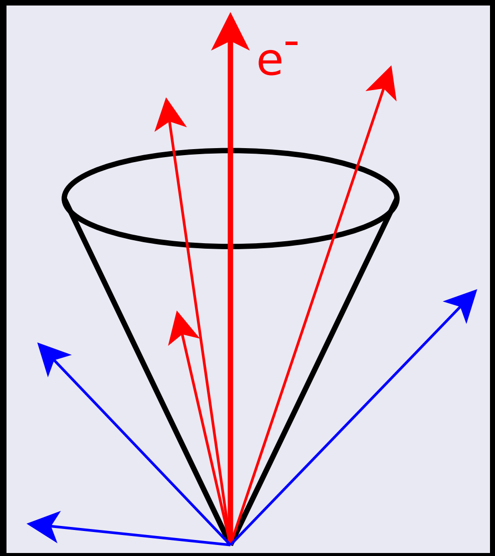
- ISF mixes different simulations
  - ➡ spend more times on important event aspects
  - ➡ dramatically reduces effects of pileup
- this idea is to do the same for tracking !
  - ➡ hence elaborate tracking for regions of interest (RoI)
    - best performance for physics objects costs CPU
  - ➡ fast tracking for underlying event and pileup
    - good enough for primary vertexing and for particle flow / jet corrections
- we do this successfully since 2012 (!)
  - ➡ calorimeter seeded brem. recovery for electrons
  - ➡ GSF later in e/gamma reconstruction





- Run-2 will have seeded TRT BackTracking
  - ➡ only reconstruction high-p_T photon conversion tracks

The ISF Idea for Tracking ?



- how could this look like ?
  - ➡ event "background":
    - extreme idea: FTK for data, truth tracking for MC + tuning ?
    - less extreme: faster tracking algorithms, compromising on performance
  - ➡ event "signal":
    - current NewTracking for regions of interests (RoIs)
    - keep electron brem. recovery
    - back tracking for conversion recovery in EM RoIs
- issues with this approach ?
  - ➡ analysis: similar complication to ISF mixed simulation
    - analysis will need to handle fast and full reconstruction objects in event
  - ➡ tracking: inside/outside RoI cone effects
    - ambiguity resolution of full tracking in RoI with fast tracking outside
  - ➡ pileup corrections for jets (including particle flow) and MET
    - requires full event reconstruction, compromise on tracking performance ?
- as well opportunities for performance optimisation !

Studies towards ISF Idea for Tracking

- "self-seeded" tracking strategy
  - ➡ variant of "Run-2" tracking setup
  - ➡ after SSS+1 candidate finding, do a z-vertex scan (like before)
  - ➡ new: find 8 vertices with largest multiplicity and ΣpT
    - restrict PPP+1, PPS+1, PSS+1 to those 8 vertices !

- significant CPU and performance gains at <μ>=140

ttbar with pileup, 25 nsec

	40 pileup		140 pileup	
seeding	efficiency	CPU	efficiency	CPU
"Run-1"	94.0%	9.5 sec	59.2%	73 sec
"Run-2"	94.2%	4.7 sec	80.4%	89 sec
"8 vertices"	94.8%	2.7 sec	82.0%	43 sec

Igor Gavrilenko, CPU on local machine

"technical efficiency" defined as efficiency to find tracks from signal event with correct hits, based on truth

- ➡ "Run-2" setup uses extra CPU at 140 pileup to recover efficiency !
- ➡ "8 vertices" would even be better for 40 pileup, but this is ttbar
  - study physics performance implications before putting it into production
- ➡ final HL-LHC setup will probably not be "self-seeded" by tracker only

Conclusions ...

- well, too early to conclude on tracking for high pileup
  - ➡ need R&D now, if we want something radically better for Phase-2 (HL-LHC)
- software technology, SIMD, cache pinning...
  - ➡ LS-1 software upgrades have shown it helps
  - ➡ at the cost of making the software more difficult to write
- tuning of algorithm strategy
  - ➡ as well, LS-1 upgrades demonstrated the potential
- multi-threading and massively parallel tracking
  - ➡ in my view it remains to be seen which role GPUs may play
  - ➡ relevant when memory/core is becoming the main issue
    - will make software even more difficult to write
  - ➡ requires to change track reconstruction strategy to avoid Armdahl's law
- need new ideas on algorithms and tracking strategies
  - ➡ definitely