Tracking In High Pile-up.

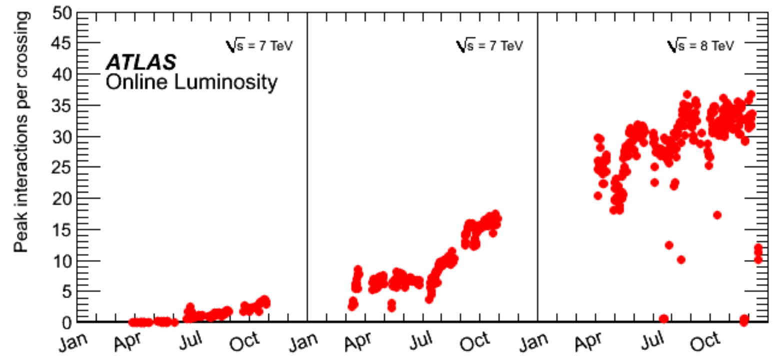
LS1 Experiences and Future Outlook

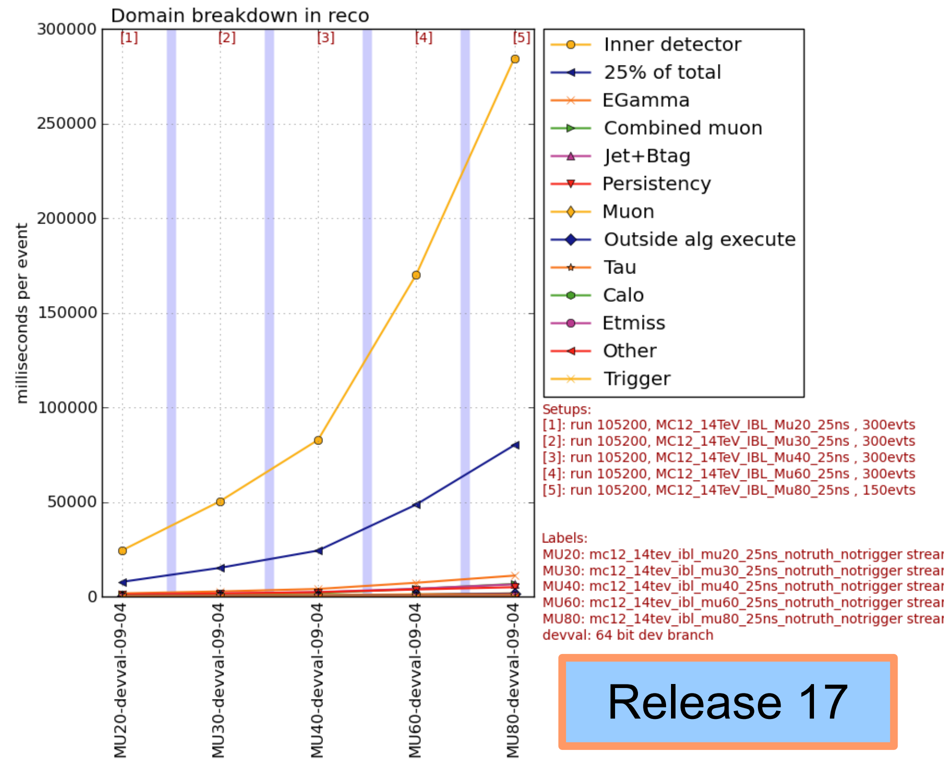
N. Styles[1]
[1]DESY,

Software TIM Meeting, Berkeley
30/09/2015

Introduction

- Reminder:
  - Were at risk of hitting a big problem with CPU after Run 1
  - Anticipated increased <mu>, and associated combinatorial increase in CPU
  - Track Reconstruction by far the biggest consumer, with worst scaling
  - Was incumbent on Tracking to make big improvements
- Large program of software improvements undertaken during LS1 to mitigate this
  - Allow reconstruction to meet goal of 1 kHz Tier-0 processing
  - ...at no cost to physics performance!





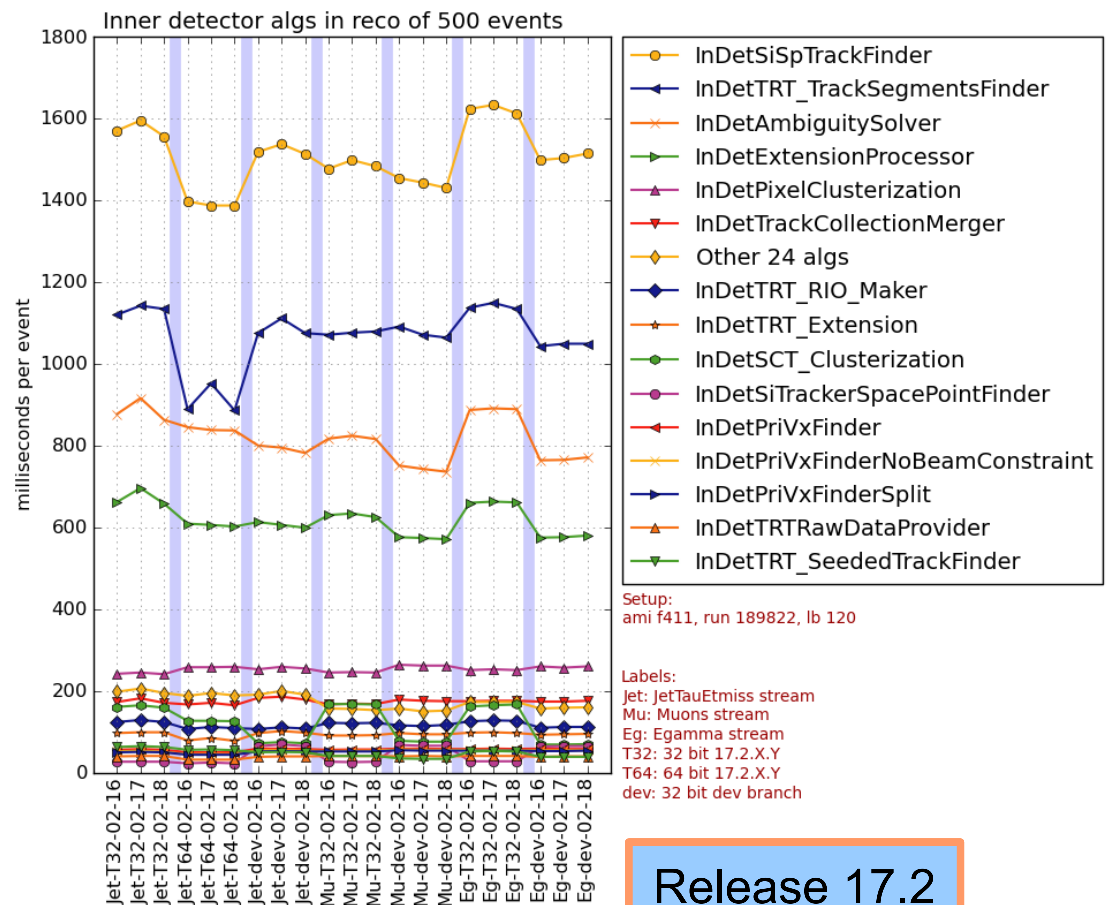
Release 17

What we knew beforehand

- “Common” (gperftools,gperftools) and ATLAS-specific tools, as well as general insight into what was running (and how) in Run 1
  - Gave directions in which to look for improvements/optimisations
- Tracking heavy user of linear algebra/matrix manipulation
  - Big gains possible from speed-ups in such operations
- Significant CPU usage in magnetic field access during Runge-Kutta propagation
  - Magnetic field service was still FORTRAN90 implementation
- Algorithmic improvements likely possible
  - Be “smarter” about what we do and when we do it
- Number of infrastructure changes bring some improvement “for free” (from tracking POV)

What we knew beforehand – Algorithm Breakdown

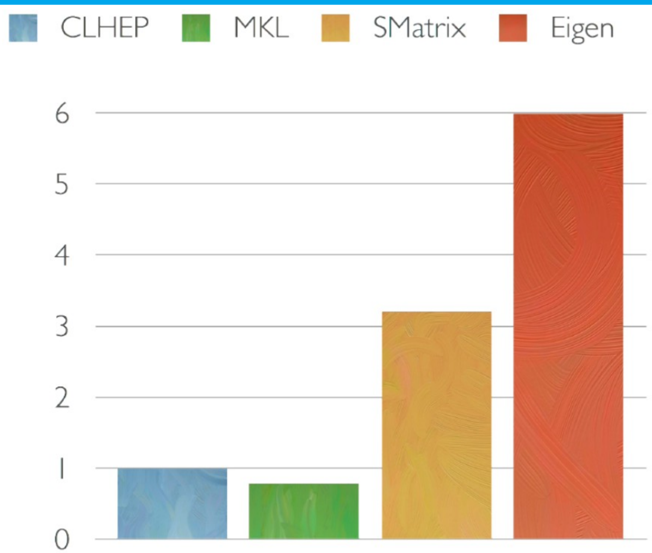
- Most time spent in Silicon Spacepoint Seeded Track Finder
  - Not surprising – main "workhorse"
  - Likewise, ambiguity and extension processing expected to be high up list
- TRT Segment finder 2nd highest
  - Part of "Back-Tracking"
  - Less clear so much time should be spent here



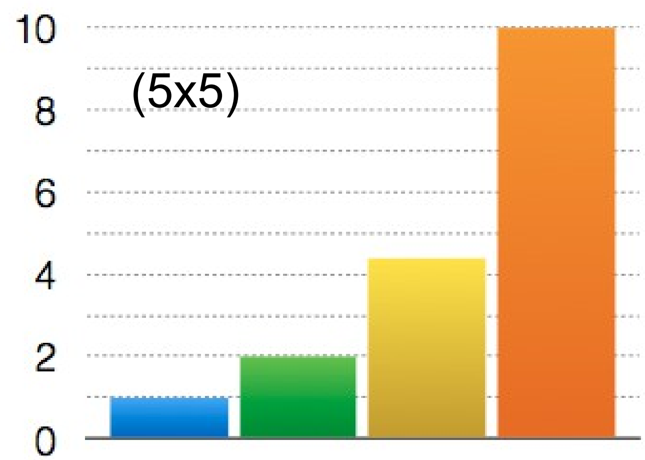
Release 17.2

Maths Library Replacement

- **>** Tested performance of alternatives to CLHEP
  - Testbed reproducing typical use cases for tracking
- **>** Replaced CLHEP with Eigen for matrix operations in track reconstruction
  - Open source, vectorised library
- **>** Required large-scale migration effort
  - Big effort from developer pool
- **>** Eigen hidden behind “Amg::” interface level
  - Helper classes for common operations not available natively in Eigen
  - Will significantly reduce overhead of any future library changes (if necessary)



Speed-up WRT CLHEP for multiplication of rectangular (3x5) matrices



Magnetic Field Updates

- Previously access to Magnetic Field information in ATLAS was through a FORTRAN90 implementation
  - This was migrated to C++
  - Code profiled and tuned during this process
  - Minimized number of unit conversions performed
- Further improvements
  - Stepwise Runge-Kutta updates can fall within same magnetic field map cell – introduced caching of position and value of last call
  - Addition of approximate, φ-symmetric map for faster access when full detail is not required
- Resulted in a significant overall speed-up in Magnetic Field Access
  - Factor >2 improvement over old implementation for a typical access pattern

Algorithmic Updates – Track Seeding

- Addition of IBL allows seed confirmation with 4th hit
  - increases seed purity
  - Reduce time spent processing track candidates that will not eventually be used
- Introduction of 'Z boundary seeding'
  - Fast 1D vertexing used to set allowed z range of seeds
- Overall >50% improvement with no efficiency loss

Fraction of seed triplets resulting in a “good” track candidate

Pile-up	PPP	PPS	PSS	SSS
0	57%	26%	29%	66%
40	17%	6%	5%	35%

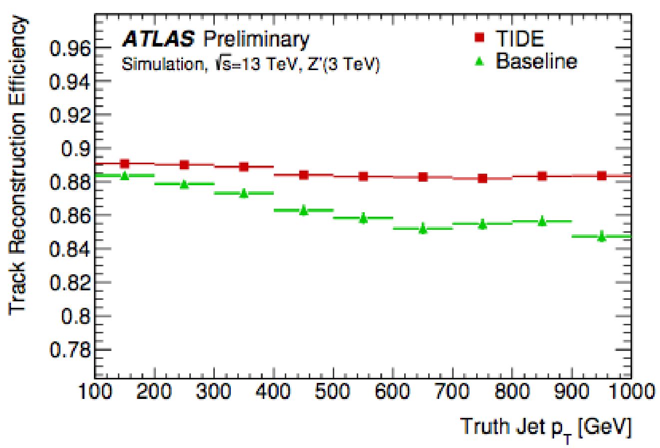
Pile-up	PPP + I	PPS + I	PSS + I	SSS + I
0	79%	53%	52%	86%
40	39%	8%	16%	70%

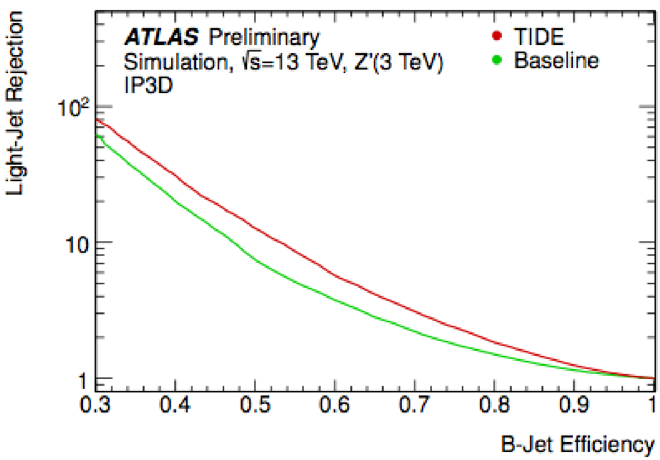
Event reconstruction time for tt at <mu>=40 on local machine

Strategy	Efficiency	CPU time
Run 1	94.0 %	9.5 sec
Run 2	94.2 %	4.7 sec

Further Algorithmic Updates

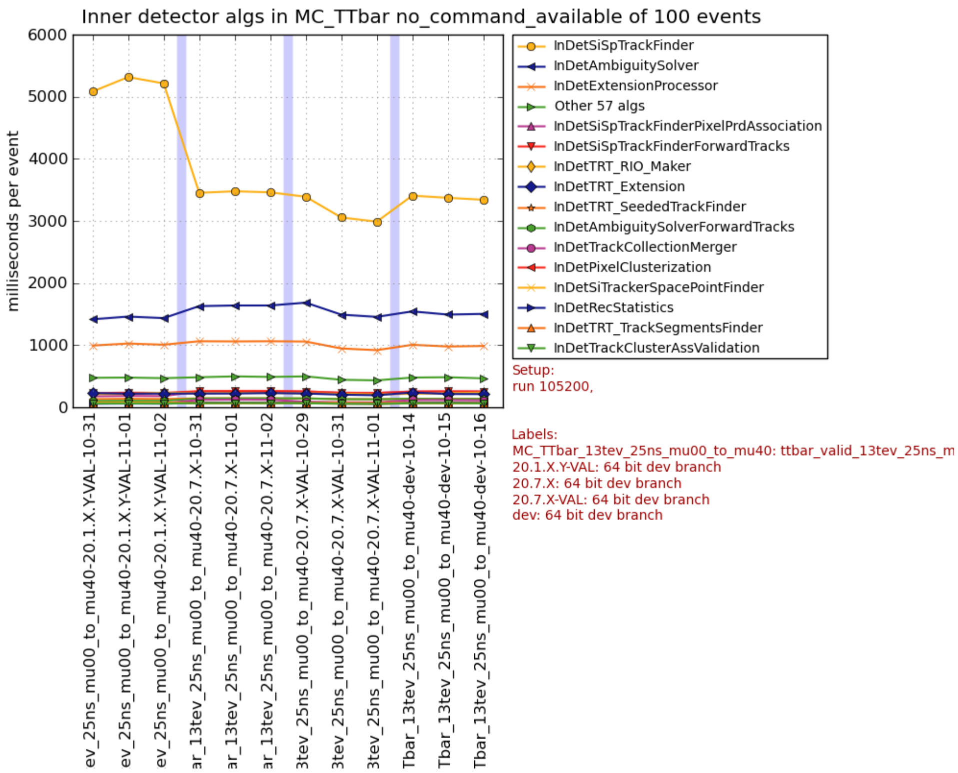
- Calorimeter-seeded back-tracking
  - TRT-seeded tracks primarily of interest for eGamma
  - Do not run back-tracking unless there is a seed calorimeter cluster
- Clustering & Ambiguity solving updates for 'Tracking In Dense Environments'
  - NN-based splitting of clusters from multiple tracks
  - For run 2, only run during ambiguity solving, for clusters on track
  - Further tuning of parameters to improve performance esp. in high-p_T jet cores
  - 10% CPU saving on top of significant performance improvements

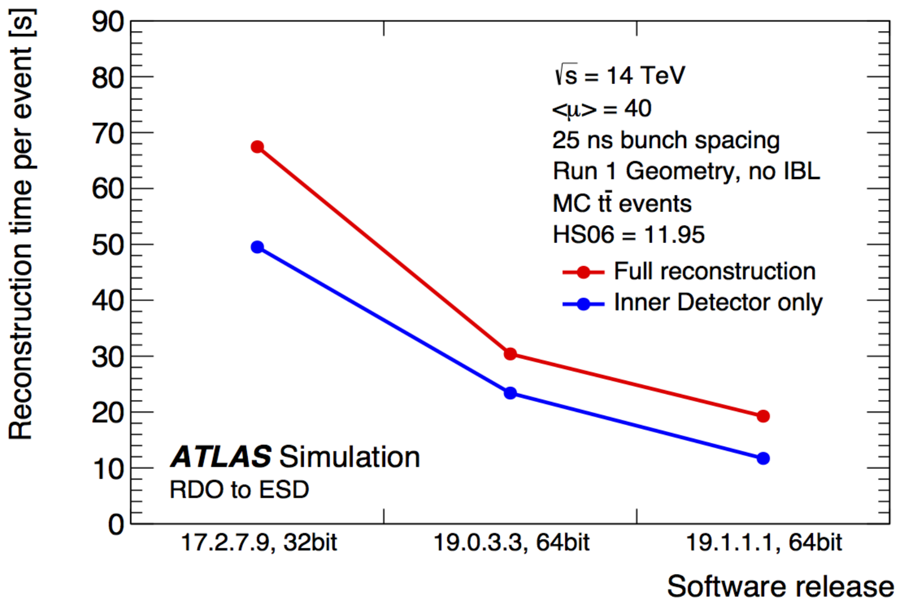




Results of LS1 Improvements

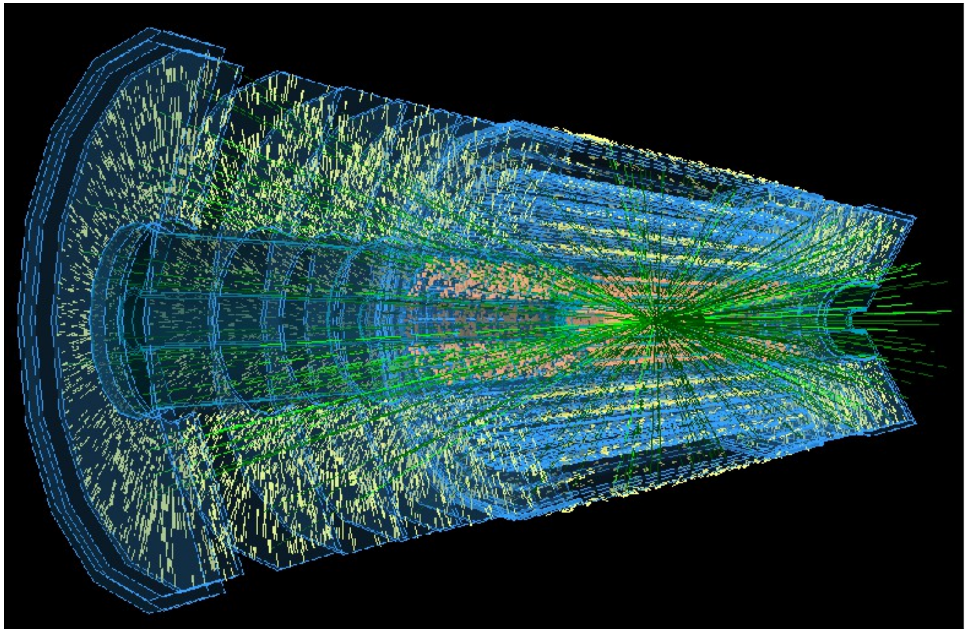
- Overall, LS1 improvements brought factor ~4 reduction in CPU usage
  - Allowed goal of 1 kHz tier-0 processing
- Further improvements from 20.1 → 20.7
  - Coming mostly from detailed optimizations of Si Track finding
  - i.e. not coming from technical updates, but rather through deep understanding and study of algorithm





Looking towards the future

- Future plans include data-taking with <mu>~80 and <mu> up to 200 following HL-LHC Upgrade
  - Large increases compared to that between Run 1 and Run 2
- Have not solved the problem of <mu> scaling of CPU time in reconstruction
  - Can still expect big increases due to increased combinatorics to deal with
- Cannot just turn the same handles again and again to win back CPU



- HL-LHC will also come together with new Inner Tracker (ITK)
  - Optimisation for a different layout, with different technologies (i.e. silicon only, no TRT)

What does HL-LHC ITk Reconstruction currently look like

InDetSCT_Clusterization:Execute	INFO Time User : Tot= 6.64 [s] Ave/Min/Max= 266(+- 33.7)/ 229/ 373 [ms] #= 25
cObjR_InDetSimDataCollection#PixelSDO_Map	INFO Time User : Tot= 6.95 [s] Ave/Min/Max= 278(+- 38.3)/ 199/ 414 [ms] #= 25
InDetSiTrackerSpacePointFinder:Execute	INFO Time User : Tot= 9.38 [s] Ave/Min/Max= 375(+- 49.4)/ 321/ 544 [ms] #= 25
cObj_InDetSimDataCollection#PixelSDO_Map	INFO Time User : Tot= 10.8 [s] Ave/Min/Max= 432(+- 61.9)/ 306/ 640 [ms] #= 25
SiSPSeededSLHCTracksDetailedTruthMaker:Execute	INFO Time User : Tot= 11.9 [s] Ave/Min/Max= 478(+- 119)/ 301/ 862 [ms] #= 25
InDetTrackClusterAssValidation:Execute	INFO Time User : Tot= 13.9 [s] Ave/Min/Max=0.558(+-0.0816)/0.416/0.742 [s] #= 25
nDetPixelClusterization:Execute	INFO Time User : Tot= 16.4 [s] Ave/Min/Max=0.655(+-0.136)/0.496/ 1.13 [s] #= 25
InDetPRD_MultiTruthMakerSi:Execute	INFO Time User : Tot= 20.6 [s] Ave/Min/Max=0.822(+-0.119)/0.566/ 1.16 [s] #= 25
InDetRecStatistics:Execute	INFO Time User : Tot= 23.5 [s] Ave/Min/Max=0.939(+-0.254)/0.552/ 1.64 [s] #= 25
commitOutput	INFO Time User : Tot= 40 [s] Ave/Min/Max= 1.54(+-0.349)/0.001/ 2.05 [s] #= 26
StreamESD:Execute	INFO Time User : Tot= 51.5 [s] Ave/Min/Max= 2.06(+-0.882)/ 1.57/ 6.27 [s] #= 25
InDetAmbiguitySolverForwardSLHCTracks:Execute	INFO Time User : Tot= 55.3 [s] Ave/Min/Max= 2.21(+-0.346)/ 1.44/ 3.03 [s] #= 25
InDetAmbiguitySolverSLHC:Execute	INFO Time User : Tot= 8.66[min] Ave/Min/Max= 20.8(+- 5.21)/ 12.4/ 34.9 [s] #= 25
InDetSiSpTrackFinderForwardSLHCTracks:Execute	INFO Time User : Tot= 25.8[min] Ave/Min/Max= 61.8(+- 9.84)/ 38.1/ 85.3 [s] #= 25
InDetSiSpTrackFinderSLHC:Execute	**INFO Time User : Tot= 75.2[min] Ave/Min/Max= 181(+- 38.1)/ 117/ 294 [s] #= 25**

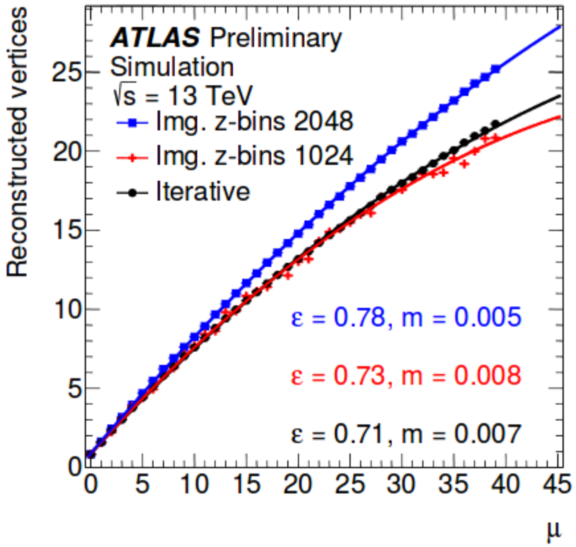
- Algorithm timing breakdown for tt events with <mu>=200
  - Ran ITK only in 20.3.1 – all other detectors switched off
- Run on random lxplus node, so not to be taken as absolute
  - Give some idea of ballpark figures
- Items in red only relevant for Monte Carlo
- ITK reconstruction is quite close to current track reconstruction
  - Many optimisations which could be made...

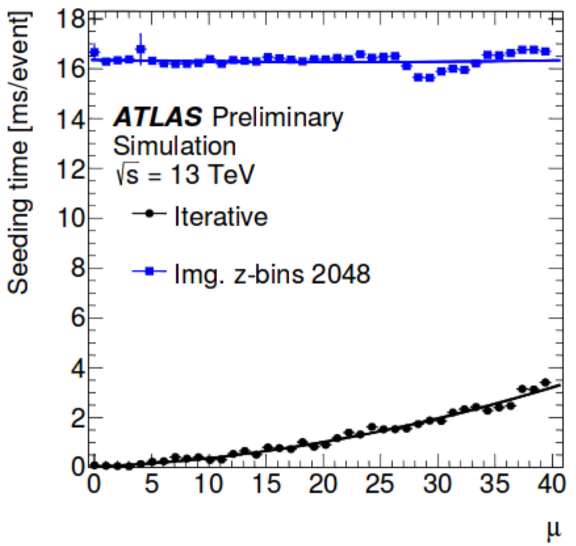
What Can/Should be Done at this point in time?

- Does it make any sense to think too hard about technical performance of ITK reconstruction now?
  - Detector layout not yet finalised – can have an influence. E.g. Layout-specific Track Seeding tunings may give advantages over using version optimised for current ID
  - ITK-specific software developments are mostly interested in improving physics performance of reconstruction – currently first priority
  - Currently only makes up a small fraction of production jobs
- However...
  - We ARE running ITK sim/digi/reco, and will in future be running more – don't want to waste resources unnecessarily if there are improvements that can be made easily
  - Aspects related to design choices in future framework may be best implemented as soon as possible
  - Fitting within the available budget will be a big challenge
- ITK software should be kept up-to-date with latest developments
  - In past has lagged behind due to specific needs or different timescales compared to general ATLAS developments

Interesting Example from Vertexing

- New vertex seeding algorithm has been in development for some time
  - Available, but not yet default, in 20.7
  - Based on ray-tracing/back projection techniques based on medical imaging techniques
  - ATL-PHYS-PUB-2015-008
- Heavy CPU overhead...
  - ...but much better scaling with <mu> - approximately flat (at least for relatively low <mu>)
  - Likely that optimization will cause 'cross-over' point to come at lower <mu>
- Vertex seeding not currently a heavy CPU consumer overall
  - However, perhaps an interesting illustrative example – trading heavier 'constant term' for better scaling





Summary

- Inner Detector Tracking suffers from significantly larger CPU overheads as pile-up increases
- Wide-ranging program of updates and optimizations during LS1 were undertaken to mitigate this
  - Reached target for Run 2
- Pile-up will continue to increase in run 3 and beyond
  - Cannot rely on just turning the same handles again
  - Andi has given overview of where/how multithreading and other aspects of parallelism can help
- Perhaps too early to start detailed optimizations of algorithms for new detector layout...
  - ...but not too early to start things about broader, general strategies for fighting against pile-up scaling
  - ...Nor for thinking about how best to operate within AthenaMT

Some random thoughts

- Are there more 'handles' we can exploit in events
  - A la what is done for TRT-seeded back tracking or brem recovery, to only run costly algorithms when strictly necessary
  - Could perhaps be compatible with use of Event Views as described by Ben yesterday?
- Tracking is generally known to be “not easily parallelisable”
  - Based around early candidate rejection – inherently serial
  - Is now the time to start thinking about what implications would be of approaches that sacrifice (some of) the early rejection power for being more amenable to parallelisation?
  - Requires going back to drawing board
  - If starting now, still time to be in good shape for HL-LHC?