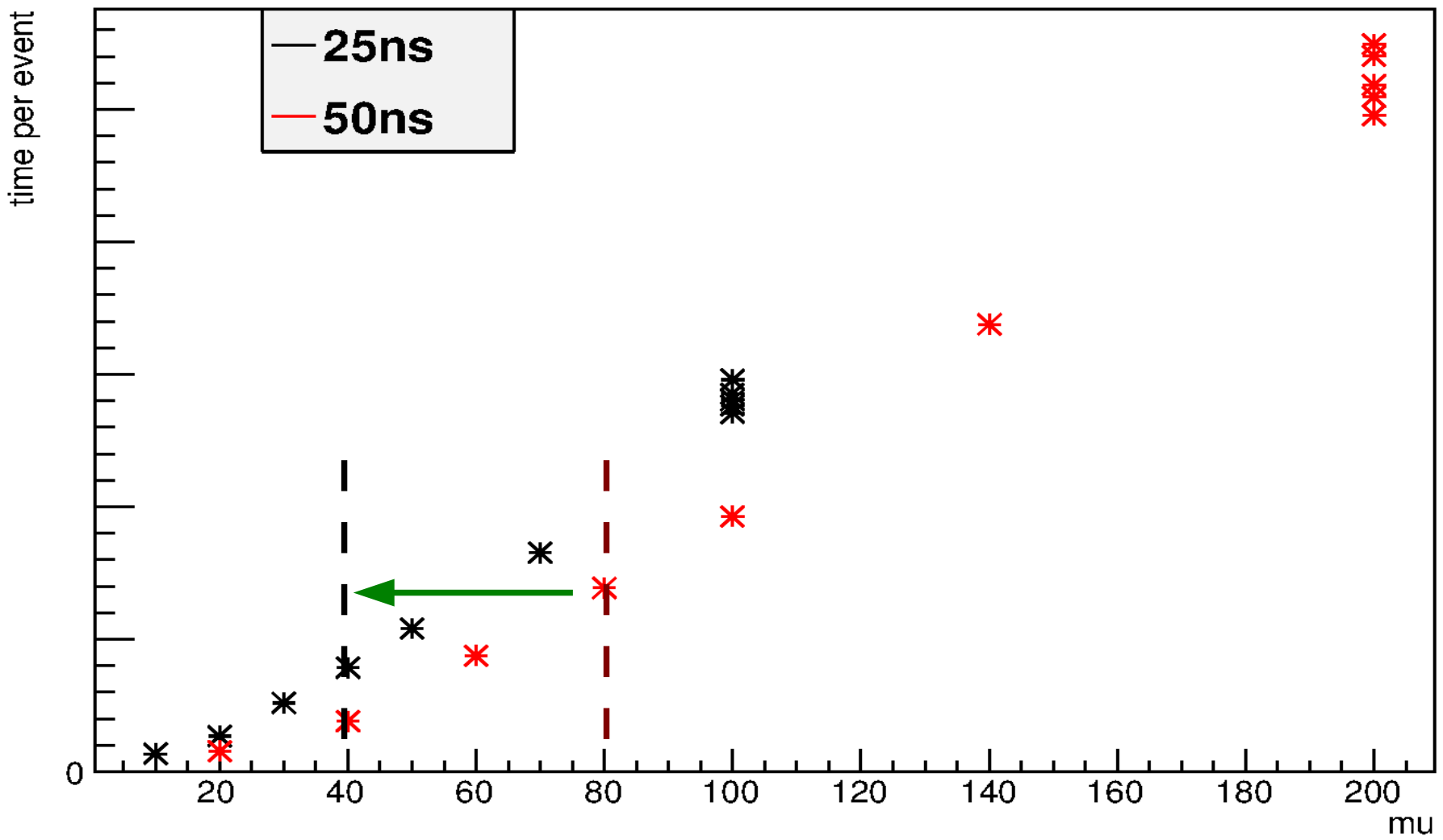
ATLAS Offline Software: Plans for LS1

Rolf Seuster (TRIUMF)
February 2013
Concurrency WS, Chicago

ATLAS data taking after LS1

- EventFilter output rate increase to ~1kHz
  - compared to 400/150Hz prompt/delayed now
- different bunch spacing (25/50ns) will give big differences in in-time pileup – big differences in CPU time/event expected
  - strong arguments from all experiments for 25ns spacing
- higher CM energy will slightly increase particle multiplicity -> even higher combinatorics
- software must gain several factors in speed !!

CPU for 25/50ns bunchspacing



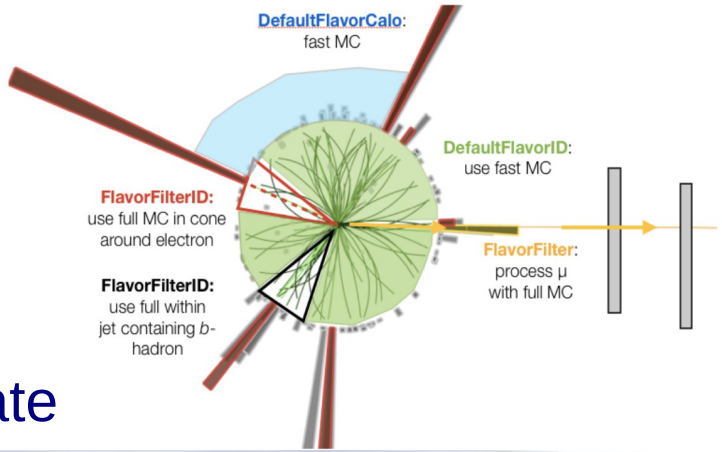
Note: private samples mimicking official jobs, misconfiguration almost guaranteed !
Strong preference towards 25 ns !
Big savings in CPU time at 25ns than 50ns for corresponding luminosity.
Note: release optimized for 50 ns, never looked seriously at 25 ns !

will now mainly focus on CPU improvements

- Event Data Model and analysis model improvements still under discussion to optimize utilization of resources: CPU+disk
- discussion is analysis focused as most access to files is from analysis group, simulation and reconstruction reads/writes files O(1)
  - should provide recommendations by March
- not part of this, but overlap of developers Look also at implementing vectorizable data object containers (1st prototype exists)

Software Speedups

- reconstruction of data events must improve by factor of ~3 to cope with increased EF rate, higher pileup, multiplicities
  - Tier0 cannot increase capacity arbitrarily and overflow to T1 difficult in past, operational problems (e.g no easy access to patch area)
  - Grid resource for MC prod. also need to keep up (to lesser extend due to ISF) Integrated Simulation Framework
    split event into chunks and simulate each with required precision



LS1 planning: general Software

- urgent tasks to work on from ~now:
  - to utilize existing hardware more efficiently
  - auto-vectorization (AV)
    - almost (?) free lunch – biggest gain by utilizing existing and future hardware more efficiently
    - not always completely for free–redesign needed
      - sometimes, ... EDM, or only within algorithm, etc.
    - possible gain promising, some algs improve by several factors (Roberto, root fitting, ...)
      - sometimes just re-designing 'EDM' also helps
    - defintely need newer compiler than gcc43
      - gcc472, clang32: code compiles mostly OK

Expected Gain for AV

- CMS: factor of ~2.5 (incl. other improvements)
  - involves partial re-write of code to utilize AV
  - handopt. ATLAS runge-kutta propagator ~2.5
- at some point manpower will limit gain, justify cost of changes w.r.t. possible gain in speed
  - collaboration with other experiments
- won't work for all cases, e.g. if conditionals inside loop usually prevents compiler to do AV
  - might be able to manually re-write code
  - still: let compiler do most of the work !
- aim for similar gain
  (but ATLAS' baseline doesn't start at zero !)

LS1 planing: general Software cont'd

- threading 2nd hot topic
- becomes very important in future with new CPU hardware – expected shortly after LS1 (next TOCK in Intel's roadmap ??)
  - have athenaMP to save on memory, see Vakho's talk
- started to look at threading via TBB, works very well for stand alone code, currenty resolving issues with athena framework
  - hope for significant improvements in near future
- will increase participation in new GaudiHive/WhiteBoard and evaluate if (and when) we switch – coordinate with trigger !!
- if offline switchs during data taking to new framework
  - maintaining two different frameworks (trigger/offline) over long time won't work !
  - offline SW volunteers to be guinea pig to deliver well tested product to whole of ATLAS

Status of Threading

- half day workshop in March to define roadmap for framework, threading will be big part of it
- first prototypes for threaded parts of reconstruction / calibration software ready
  - tracking code using pthreads / TBB:

    shows good scaling with low number of threads
  - Calorimeter Calibration code AV + threaded
    - see this talk in concurrency forum for details

math functions

- math functions used abundanlty in ATLAS, e.g.BField calls ~7.2 mio times atan2/sincosf on first event (this years data with highest <μ> so far) and 560mio times in following events 2-100 ! In 2nd place a monitoring tool with 112 mio times calling log and tan each, 3rd BField again and 4th TrkNeuralNetworkUtils
- will cause performance degradation if used that often, full precision needed ? Faster, less precise implementations available, or use Intel's Math library (~10% in simulation) or VDT from CMS
- look at implementation with same/similar precison over limited range (η in tracking restricted to $|\eta| \lesssim 2.5$)
- more details in this talk

e.g. Magnetic Field

- big speedups already achieved by caching, clean up of call chain and math functions
- Fortran code re-written in C++ (maintainability)
  - now also AV looked at
- first tests show big speedup, up to factor 2.6 faster field lookup in test simulation job
  - reconstruction jobs use typically less field lookups, gain here to be evaluated

CLHEP Improvements

- strong interest in boosting performance of CLHEP or suitable replacement from reco
  - CLHEP itself not developed any more
  - replacement should have same or similar interface, enable auto-vectorization, not go through function call overhead (e.g. rules out BLAS), inlined as much as possible
  - developed small testbed for quick turnaround of new test implementations of improved CLHEP implementions or replacements
  - looking also at improving CLHEP

CLHEP called abundantly

Calls	Total Instr.	Avg. Instr.	Function
1428113303	27804576097	19.47	CLHEP::Hep2Vector::operator()(int) const
607376609	18629216328	30.67	HepGeom::Transform3D::operator()(int, int) const
740001697	12580419327	17.00	CLHEP::Hep3Vector::operator()(int)
3547958	10147365004	2860.06	CLHEP::operator*(CLHEP::HepMatrix const&, CLHEP::HepSymMatrix const&)
47233820	9824634560	208.00	CLHEP::HepRotation::rotateAxes(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&)
475620193	9224050932	19.39	CLHEP::Hep2Vector::operator()(int)
3245232	7425633818	2288.17	CLHEP::operator*(CLHEP::HepSymMatrix const&, CLHEP::HepMatrix const&)

- about 60% of operator* of two HepMatrices are for 3x5 times 5x3 or 5x5 and 5x5 matrix multiplications (from Graeme Steward)
- also know from where they are called
  - now need to look at code and better implementations of matrix libraries
  - covered in AV section of this WS

Summary

- due to expected LHC performance after LS1 considerable speedup for reco / sim need
- not mentioned so far: move to 64 bit will gain ~20% at cost of higher memory consumption
  - future: x32 ABI: 64 bit code with 32 bit pointers
  - RedHat reluctant to implement, hard to bootstrap, virtual machines might help here
- not mentioned in this talk:
  - needed EDM improvements incl. changes to analysis model