Digitization Update

- Digitization Updates Over The Summer
- Job Transform Changes
- Run Number/LumiBlock/Pile-up Lumi overrides
- Open Issues in 16.X.0
- Outstanding Tasks for 16.X.0
- Future plans
- Summary

Digitization Updates Since Last Software Week

- Pile-up code is being used more commonly in production jobs.
- Validation of Bunch Structure and BCID setting in pile-up Simulation almost completed using 16.0.2.Y – final few issues fixed last week.
- Job Transforms:
  - Digi_trf.py replaced csc_digi_trf.py in 16.0.0 onwards.
  - Job options fragments now use digitization job properties.
- Overriding Run Number/Lumi Block/Pile-up Lumi on an evt-by-evt basis now possible – some sub-detectors starting to provide additional conditions information so this functionality can be used.
- Progress on PileUpTools memory requirements – see later.
- Most RTT and ATN jobs updated to use transforms/more recent input files (all now use mc09 or mc10 inputs). We now also check for differences in behaviour between jobs using the most recent DBRelease and oracle.

Job Transforms Changes

Task	Old Transform	New Transform
G4sim	csc_atlasG4_trf.py	AtlasG4_trf.py
Cosmics Evgen+G4sim	csc_cosmics_sim_trf.py	AtlasG4_trf.py
Cosmics G4sim from TR input	csc_cosmicsTR_sim_trf.py	AtlasG4_trf.py
Filter HIT files	csc_filterHIT_trf.py	FilterHIT_trf.py
Merge HIT files	csc_mergeHIT_trf.py	Merging_trf.py
Digitization	csc_digi_trf.py	Digi_trf.py
Pile-up Digitization	csc_digi_trf.py	Digi_trf.py
G4sim+Digitization	csc_simul_trf .py	Sim_trf.py
G4sim+Pile-up Digitization	Not Possible	Sim_trf.py
Cosmics Sim+Digitization	Not Possible	Sim_trf.py
Sim+Digi+Reco	csc_simul_reco_trf.py	Not yet implemented

- Only new transforms available from 16.0.0 onwards.
- All non-sim/digi-specific arguments imported from PATJobTransforms.
- Used in production with 16.0.2.Y
- In the future we will implement an overall transform which will allow Sim+Digi+Reco.

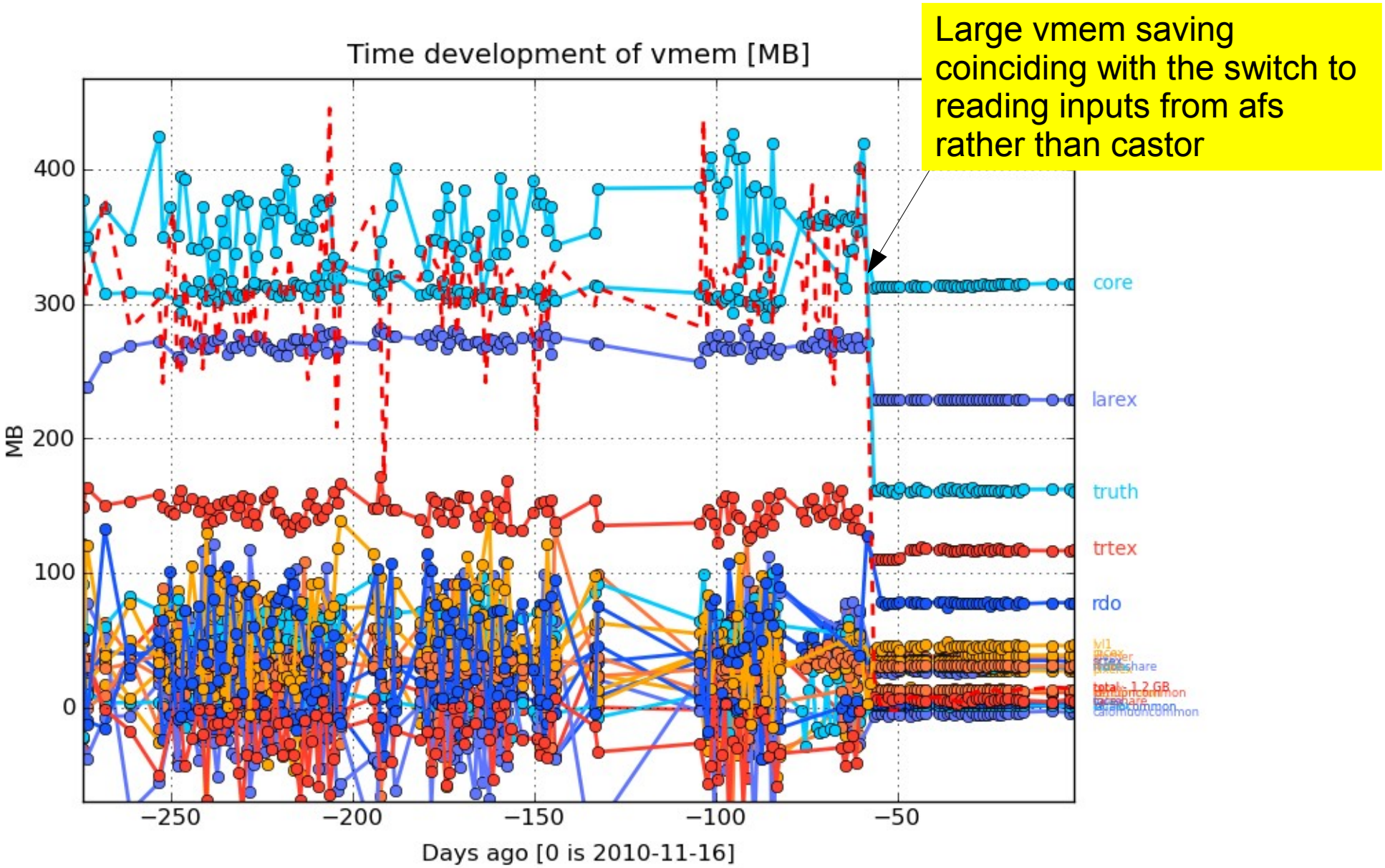
Digi_trf.py Job Options

As mentioned previously the configuration in job options files for Digi_trf.py in the 16.X.Y.Z nightlies now uses the Digitization jobproperties – just like normal athena jobs.

Example job options fragment for use with Digi_trf.py:

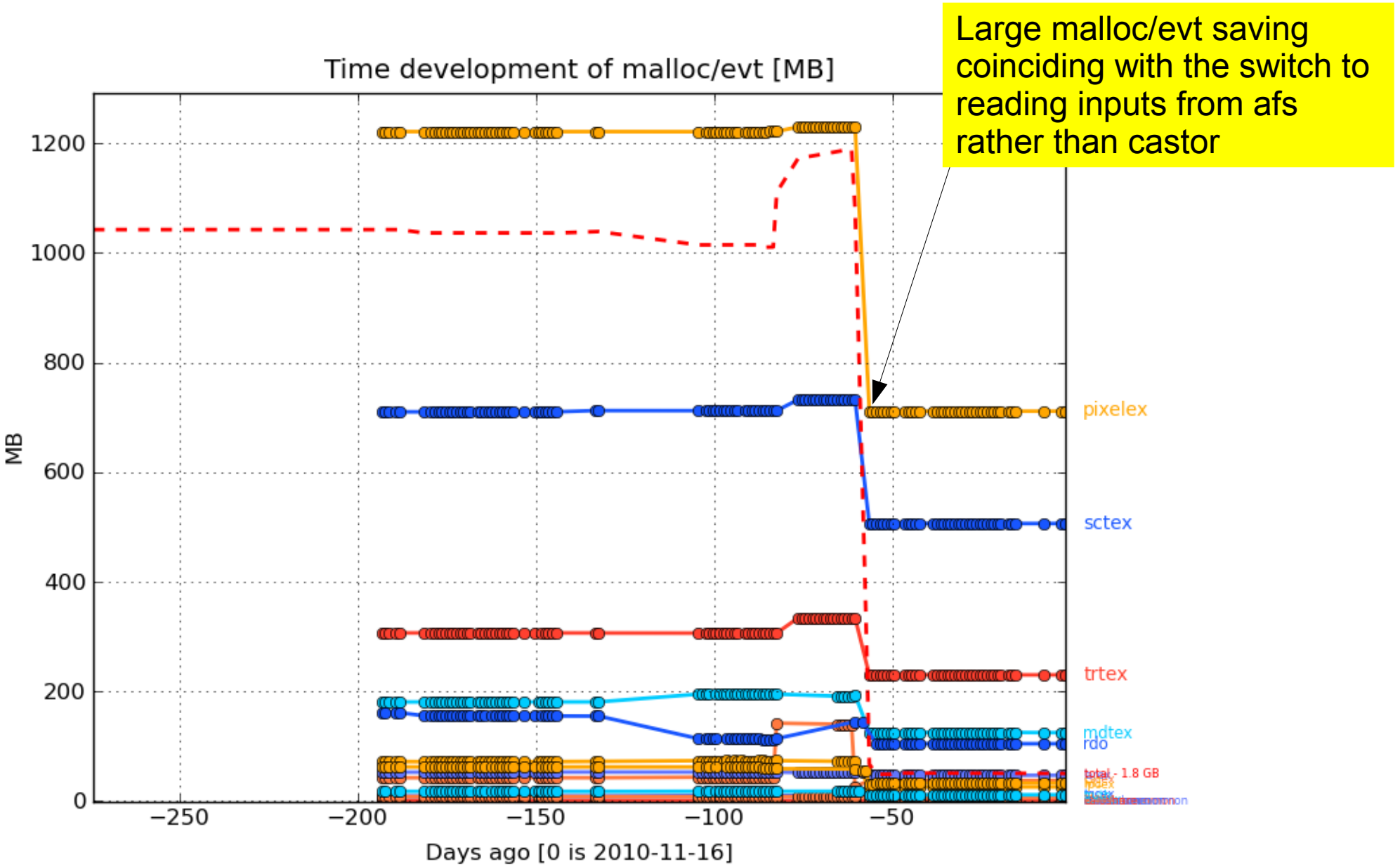
```
from Digitization.DigitizationFlags import jobproperties
jobproperties.Digitization.bunchSpacing = 25 #default
jobproperties.Digitization.initialBunchCrossing = -32 #default
jobproperties.Digitization.finalBunchCrossing = 32 #default
#N cavern should now be proportional to beam lumi, rather than bunch lumi
jobproperties.Digitization.numberOfCavern = 2
jobproperties.Digitization.numberOfCollisions = 4.6 #<No. ND minbias>
jobproperties.Digitization.numberOfBeamHalo = 0.05 #<No. beam halo>
jobproperties.Digitization.numberOfBeamGas = 0.0003 #<No. beam gas>
jobproperties.Digitization.numberOfSDMinBias = 1.0 #<No. SD minbias>
jobproperties.Digitization.numberOfDDMinBias = 1.0 #<No. DD minbias>
# The shortest repeating unit in the beam intensity pattern.
# If this variable is set then ArrayBM or FixedArrayBM will be used.
jobproperties.Digitization.BeamIntensityPattern =
[1.0,1.0,1.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0]
# Specify the position in the pattern, where the signal event occurred.
# If this is specified then FixedArrayBM will be used.
jobproperties.Digitization.FixedT0BunchCrossing=1
# By default the cavern background will ignore the beam intensity pattern,
# but this can be altered for debugging by setting this property to False.
jobproperties.Digitization.cavernIgnoresBeamInt=True
```

Evt Loop vmem 3.5E33 Pile-up



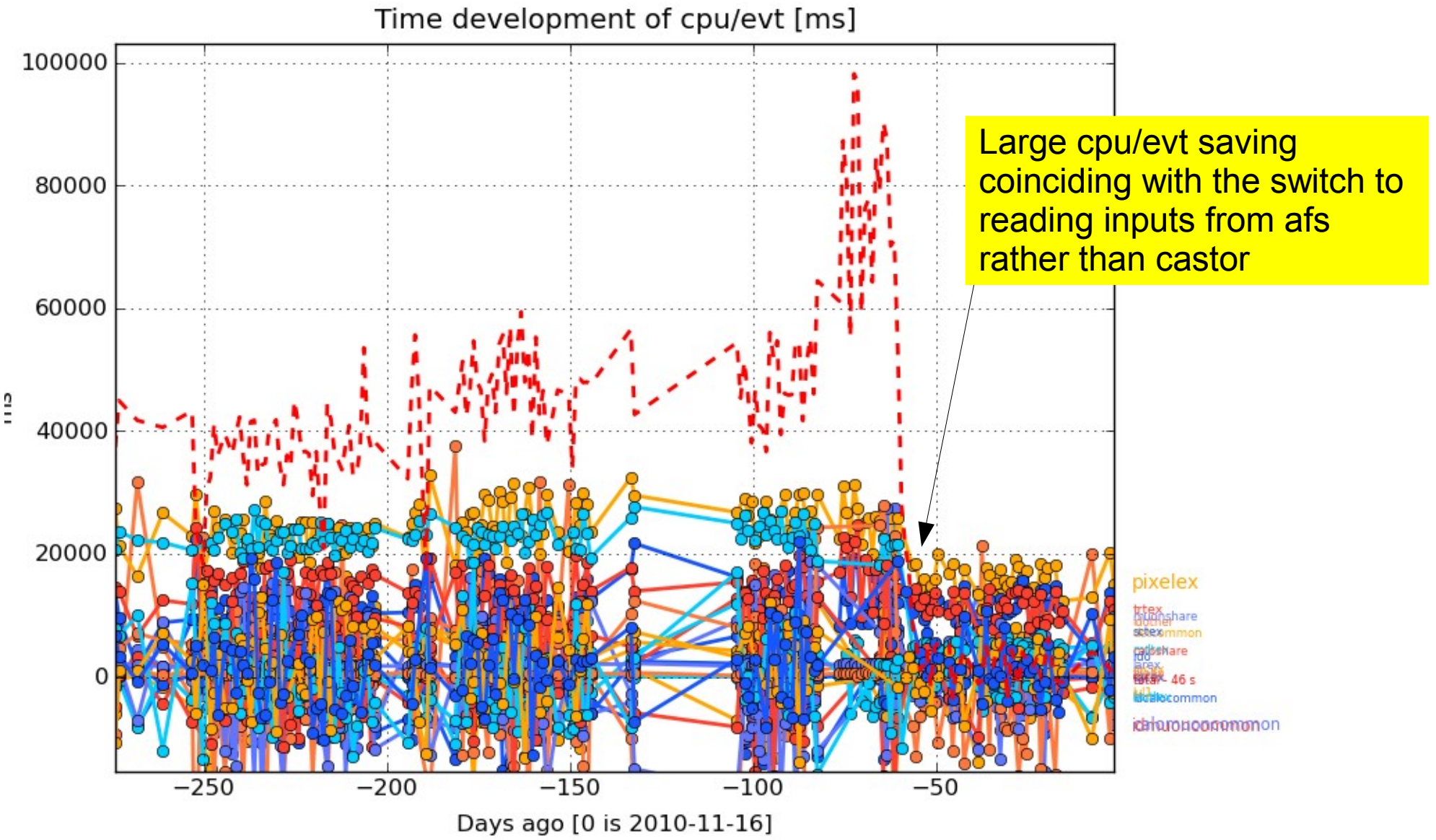
(Results from dev nightly RTT jobs.)

Evt Loop Malloc Usage 3.5E33 Pile-up



(Results from dev nightly RTT jobs.)

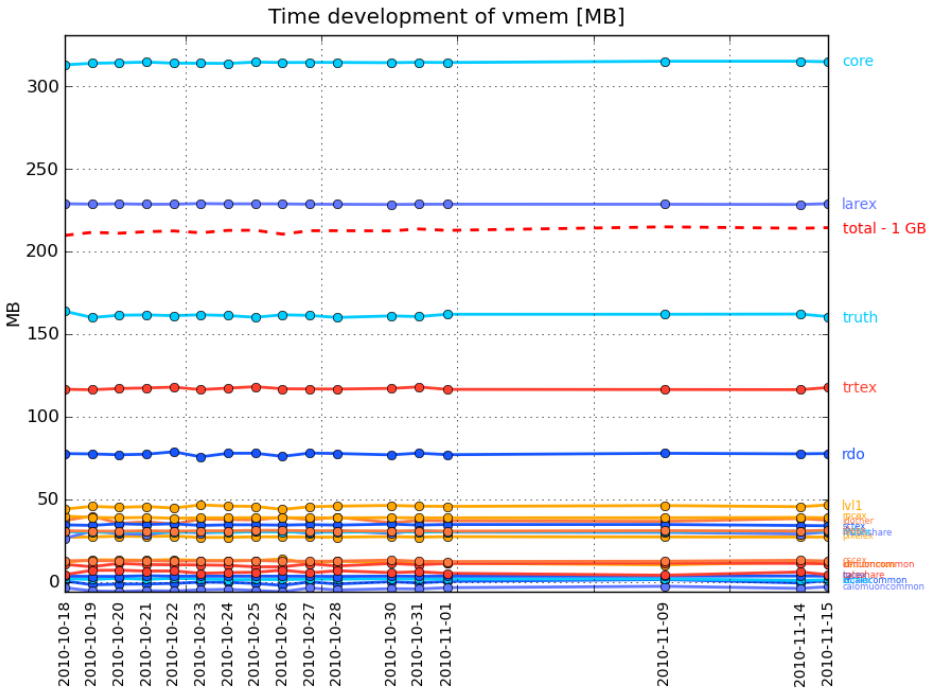
Evt Loop CPU Usage 3.5E33 Pile-up

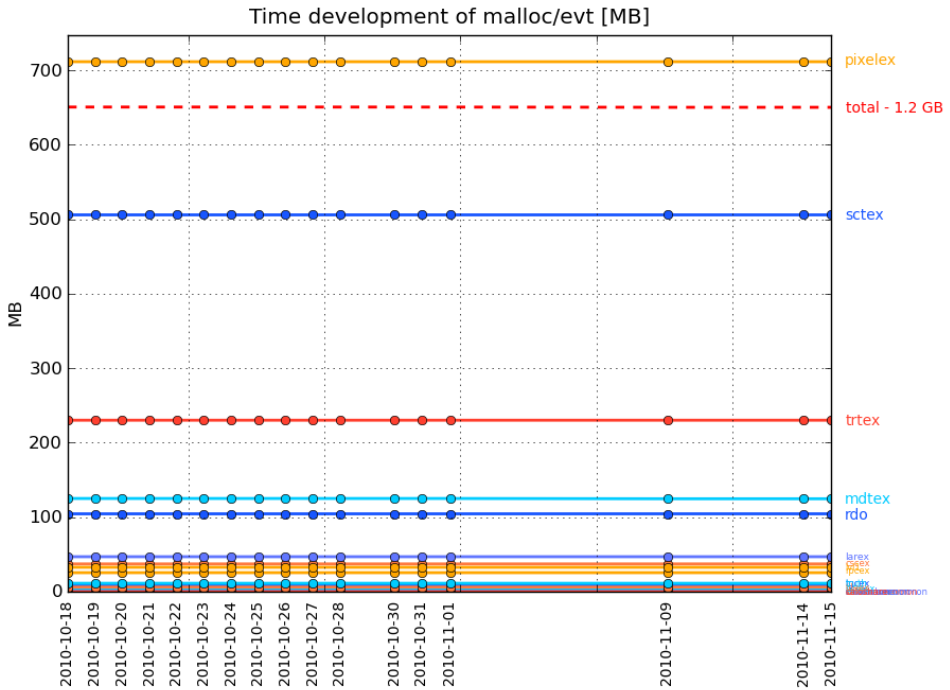


(Results from dev nightly RTT jobs.)

vmem and malloc Usage - Details

Full information here:
http://atlaspmb.cern.ch/rtt-mon/?var=vmem&build=dev&cmtconfig=i686-slc5-gcc43-opt&jobset=digipile&cache=prod

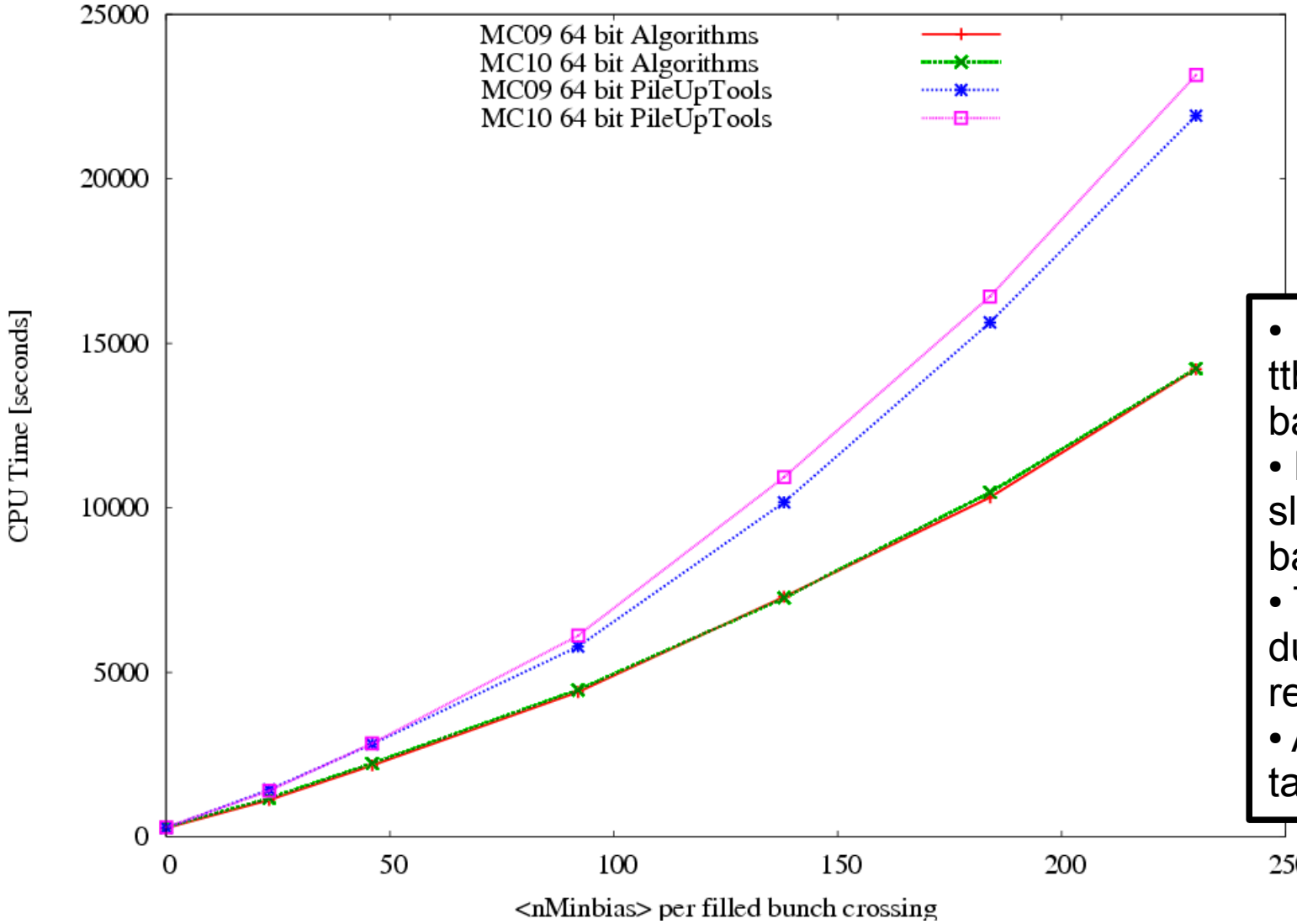




- As before, the above plots are for 3.5E33, 25ns pile-up using mc09 inputs and run as part of the dev nightly RTT.
- Digitization performance has remained constant for some-time now.
- Still waiting for validation of a new MDT Digitization Tool written by Dinos Bachas – hopefully with 16.0.2.7
- SCT developers exploring alternative digitization schemes, which may reduce the number of malloc calls.
- Suggestion that Pixels and SCT should look into using data pools.

High Luminosity Pile-up

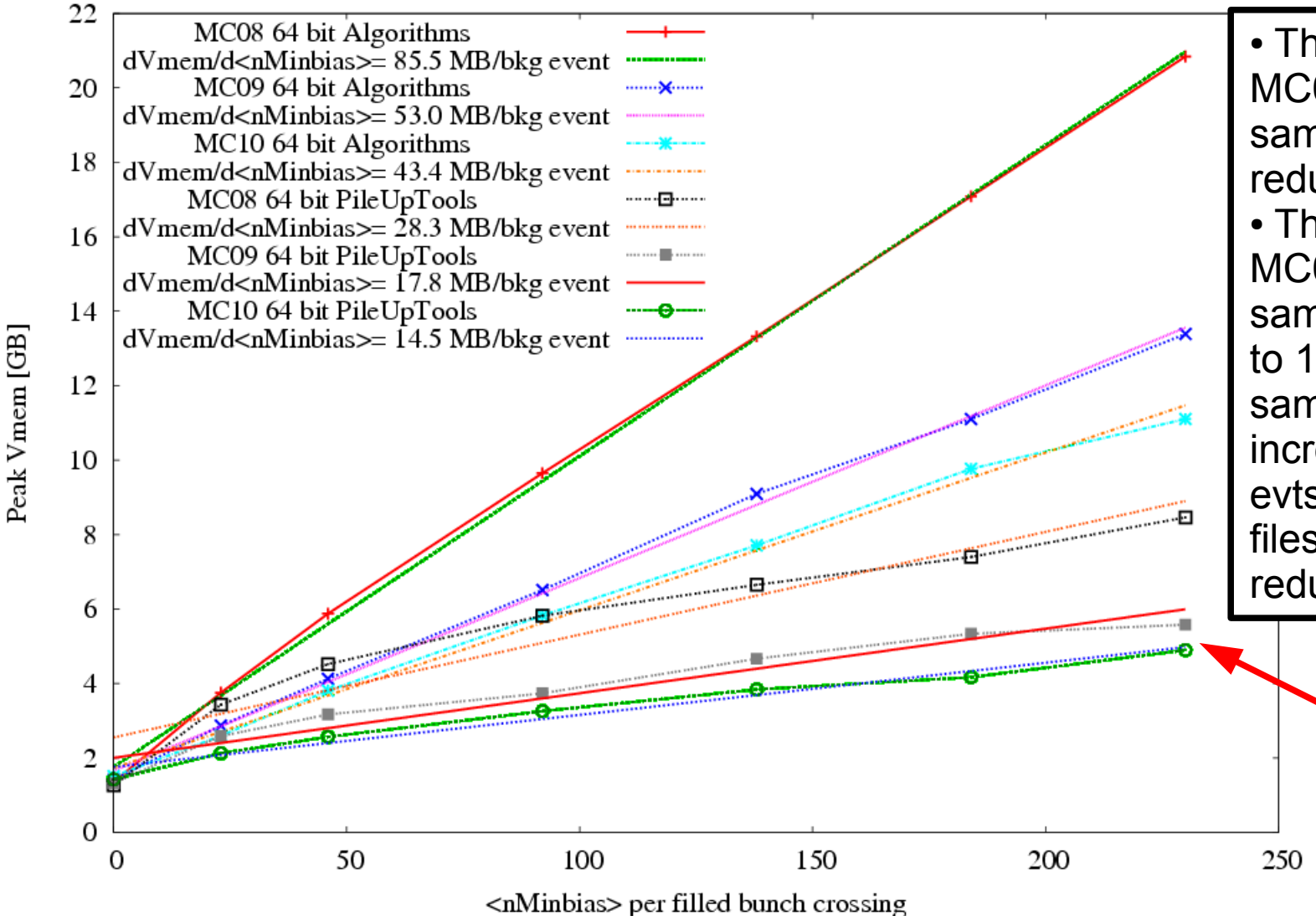
Job Time Scaling with 16.3.0 (50ns bunch spacing)



- All jobs run over 25 ttbar events on the LSF batch system.
- PileUpTools jobs are slower than Algorithm based jobs.
- This is to be expected due to the increased I/O required.
- A 25 event 5E34 job takes ~6.75 hours.

Memory Scaling with 16.3.0 (50ns bunch spacing)

- All jobs run over 25 ttbar events on the LSF batch system.



- The drop between MC08 and MC09 samples is due to a reduction in filesize/evt.
- The drop between MC09 and MC10 samples is probably due to 14TeV vs 7TeV samples and an increased number of evts/file, meaning fewer files are open and a small reduction in filesize/evt.

5E34 pile-up using PileUpTools now requires <6GB!

Open Issues In 16.X.0

- PileUpTools (task#16611)
  - Hit-by-hit comparison not available. Hope to do large scale validation with 16.0.2.7 once 64-bit athena validation is complete.
  - Solve problems with I/O memory cost (bug #62074).
- Jobs with Run Number = 0 (bug #70694 & bug #75656)
  - Usually this is the result of mis-configuration during evgen.
  - It would be really useful to have a definitive policy to point to on this though. Is MC with Run Number = 0 supported?
- MetaData IOVs in RDO files generated with 64-bit athena cannot be read by 32-bit athena. (bug #75583)

Outstanding Tasks for 16.X.0

- Task #16577 **Integration of ALFA Digitization**
  - Stand-alone code exists. Need to get G4Sim step working before integration of ALFA Digitization with athena can be tested.
- Task #16612 **Switch on LUCID Digitization by default**
  - Will implement once HITS files with simulatedDetectors MetaData are available.
- Task #16621 **New Job Transform for Cavern Background and Beam Halo Production**
  - Currently integrating BeamHaloGenerator into Evgen_trf.py
- **Integration of ZDC Digitization** (no task number yet)
  - Currently waiting for ZDC digitization code to be available in SVN.

Future Plans

- PileUpTools validation with 64-bit athena – soon?
- Re-factor PileUpTools code to share code with Algorithms where possible (on-going).
- Investigate reading in more complicated bunch structure patterns from the DB (for Release 17).
- Further improvements to job transform code – later this week.
- Update validation of pile-up (PileupRTT)
- Improve/update sub-detector Digitization RTT packages.
- Take a fresh look at areas for optimization within sub-detector code.
- Improve Digitization documentation.

Summary

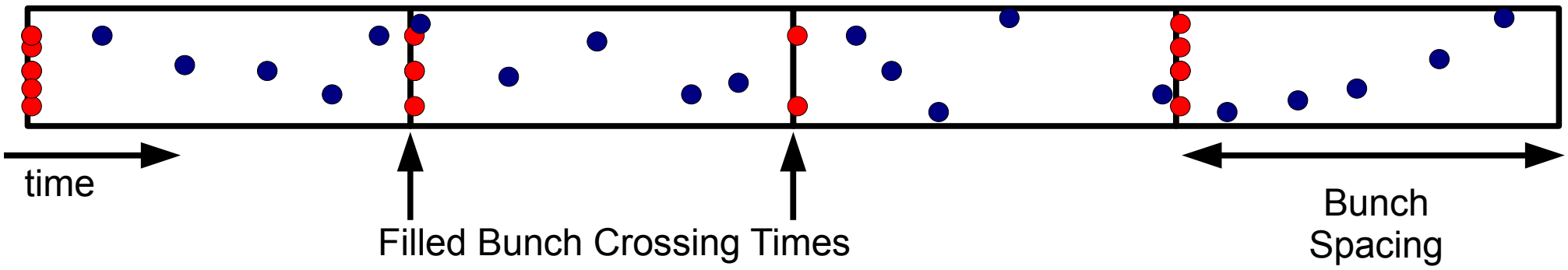
- Bunch Structure Simulation almost validated with 16.0.2.Y. Next step is to read Bunch Structure patterns from the database.
- Run Number/LumiBlock/Pile-Up Lumi override code available.
- New job transforms being used in production with 16.0.2.Y.
- Improving the robustness of nightly validation.
- HIT file EDM and compression improvements give a dramatic reduction in High Lumi Pile-up vmem requirements. It is now possible to run 5E34 pile-up on lxbatch even with the Algorithm version of the Digitization code (64-bit)! PileUpTools require less than 6GB now!
- PileUpTools ready for validation – with 64-bit athena.

Backup

Event Timing For Different Background Types

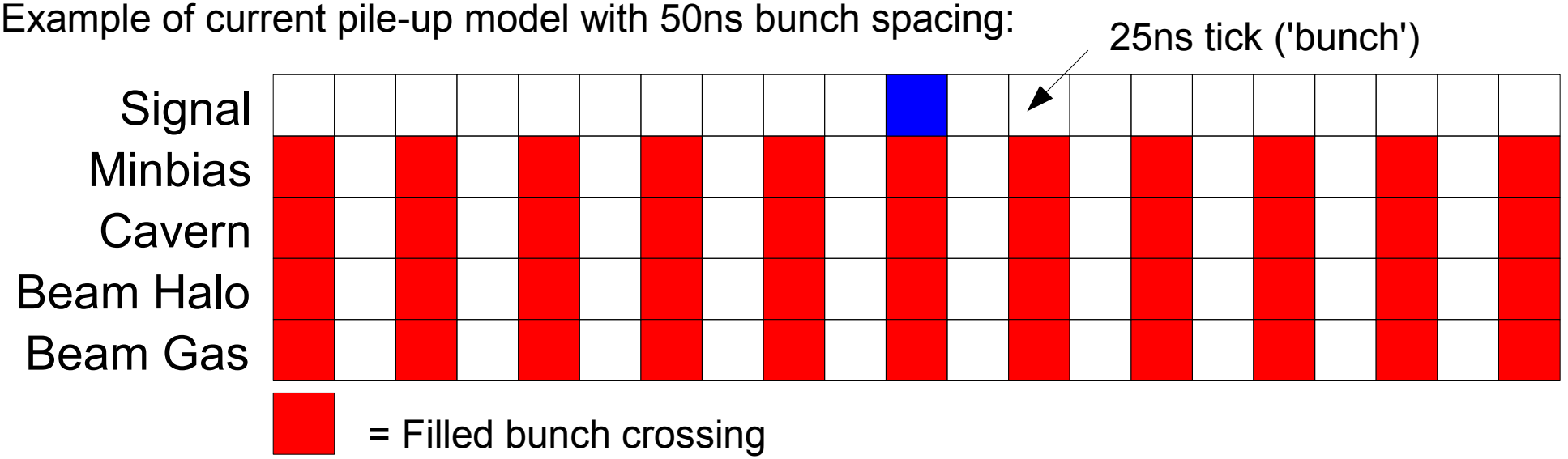
- Minimum Bias, Beam Halo, Beam Gas: All events are simulated as starting at the bunch crossing time.
- Cavern Background: Separate samples used for each bunch spacing to be simulated. Event starting times are spread over the whole bunch spacing time in order to produce a constant background.

In both cases events are then offset in time depending on the particular bunch crossing they are being used for.



Bunch Structure (I)

Example of current pile-up model with 50ns bunch spacing:



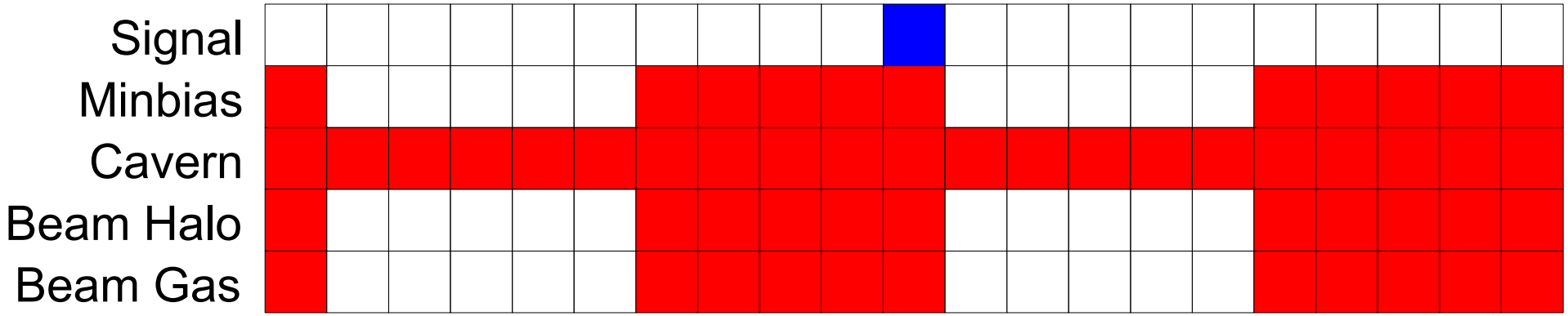
In reality the structure of filled and empty bunch crossings can be more complicated.

Here there are clear effects on the pile-up and hence detector response depending on where in the bunch train the triggering signal event occurs.

Bunch Structure (II)

Paolo Calafiura, John Chapman

From 15.8.0.3 onwards, the pile-up code will add the possibility of simulating more complicated bunch structures. E.g. 5 filled bunch crossings followed by 5 empty bunch crossings. In this case backgrounds would be added as follows:



Note how cavern background would be added at a constant rate, removing the need to simulate separate samples for each bunch spacing.
Initially the bunch structure will be specified via python job options, but eventually the idea is that the it will be read from the conditions database.
Paolo Calafiura has implemented this using the new IBeamIntensity services.
http://indico.cern.ch/conferenceDisplay.py?confId=94183
Currently there are 3 services available that use this interface:
• FlatBM – reproduces previous pile-up simulation with fixed bunch spacing (default)
• ArrayBM – signal bunch crossing is picked at random from the filled bunches in the pattern.
• FixedArrayBM – the same as ArrayBM, but the signal bunch crossing is fixed to a predefined point in the pattern.
https://svnweb.cern.ch/trac/atlasoff/browser/Control/PileUpComps/trunk/src

PileUpTools: BC by BC Pile-Up

- Provides one filled bunch crossing at a time to all interested sub-detectors.
- A single pile-up Algorithm (PileUpToolsAlg) calls an AlgTool for each sub-system. The AlgTools know the time window for which they are sensitive to bunch crossings.
- No sub-event cache. Sub-events are read as required and discarded from memory after each filled bunch crossing is processed.
- After all filled bunch-crossings have been processed. Digits/RDOs are produced from intermediate information cached locally by the sub-detector tools.
- This only helps if the local caches of intermediate information are smaller than the current cache of Hits..

Algorithm and PileUpTools Approaches to Pile-up Digitization