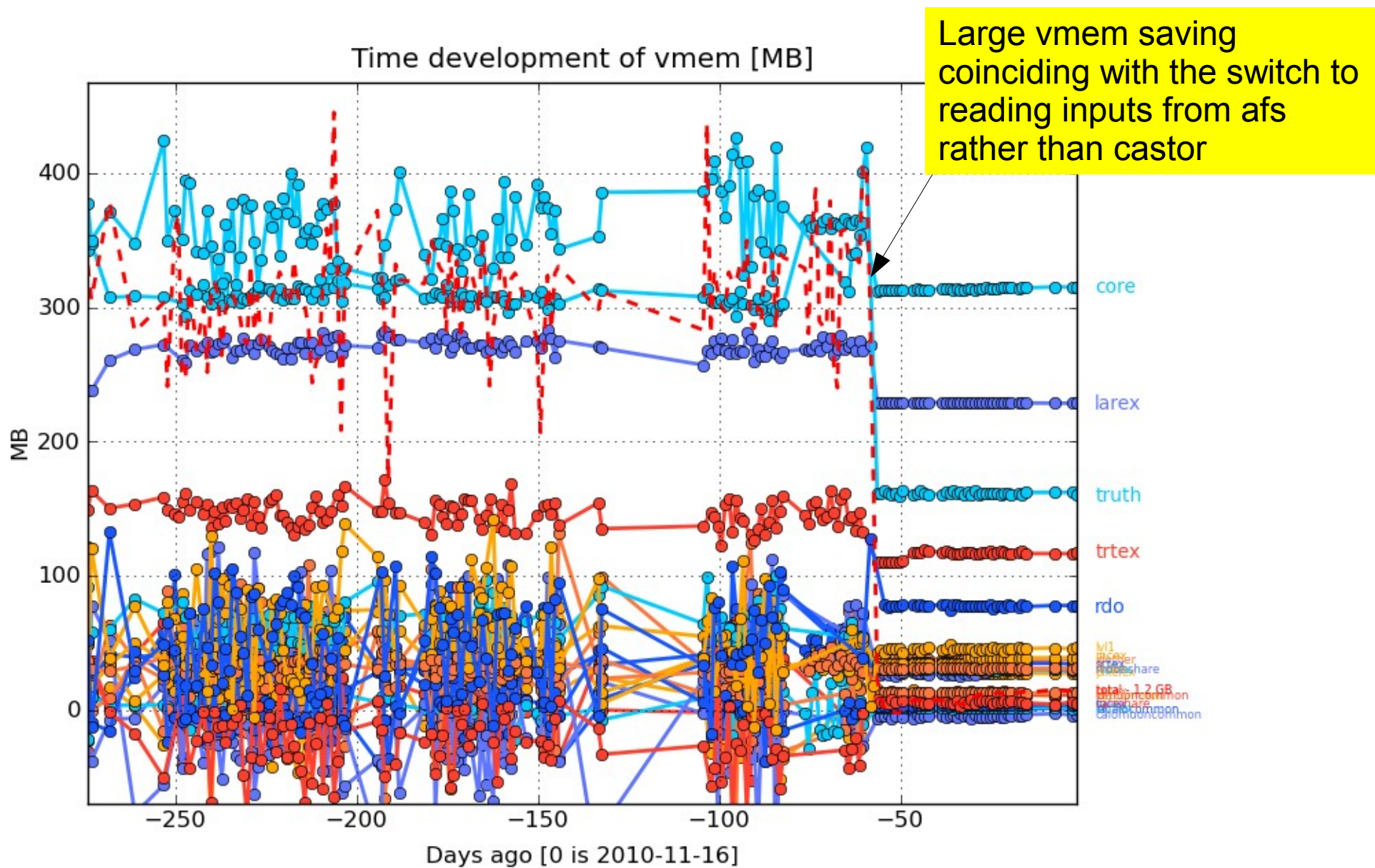


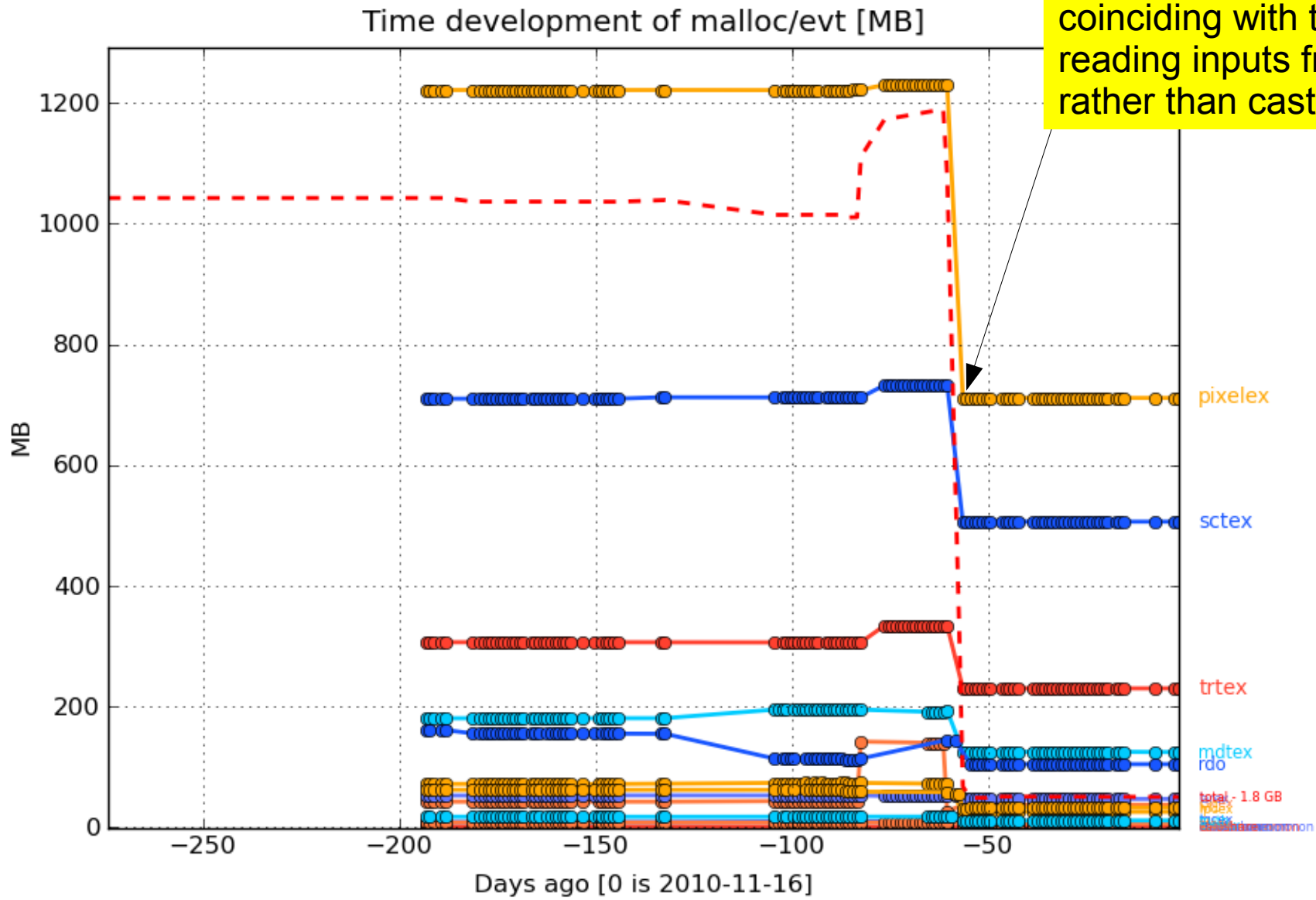
Digitization Performance Update

- Per Domain Performance breakdown with 32-bit dev build
- Plots of status High Luminosity Pile-up Status with 64-bit dev builds
- Steps Required Before Production
- Future Plans
- Summary

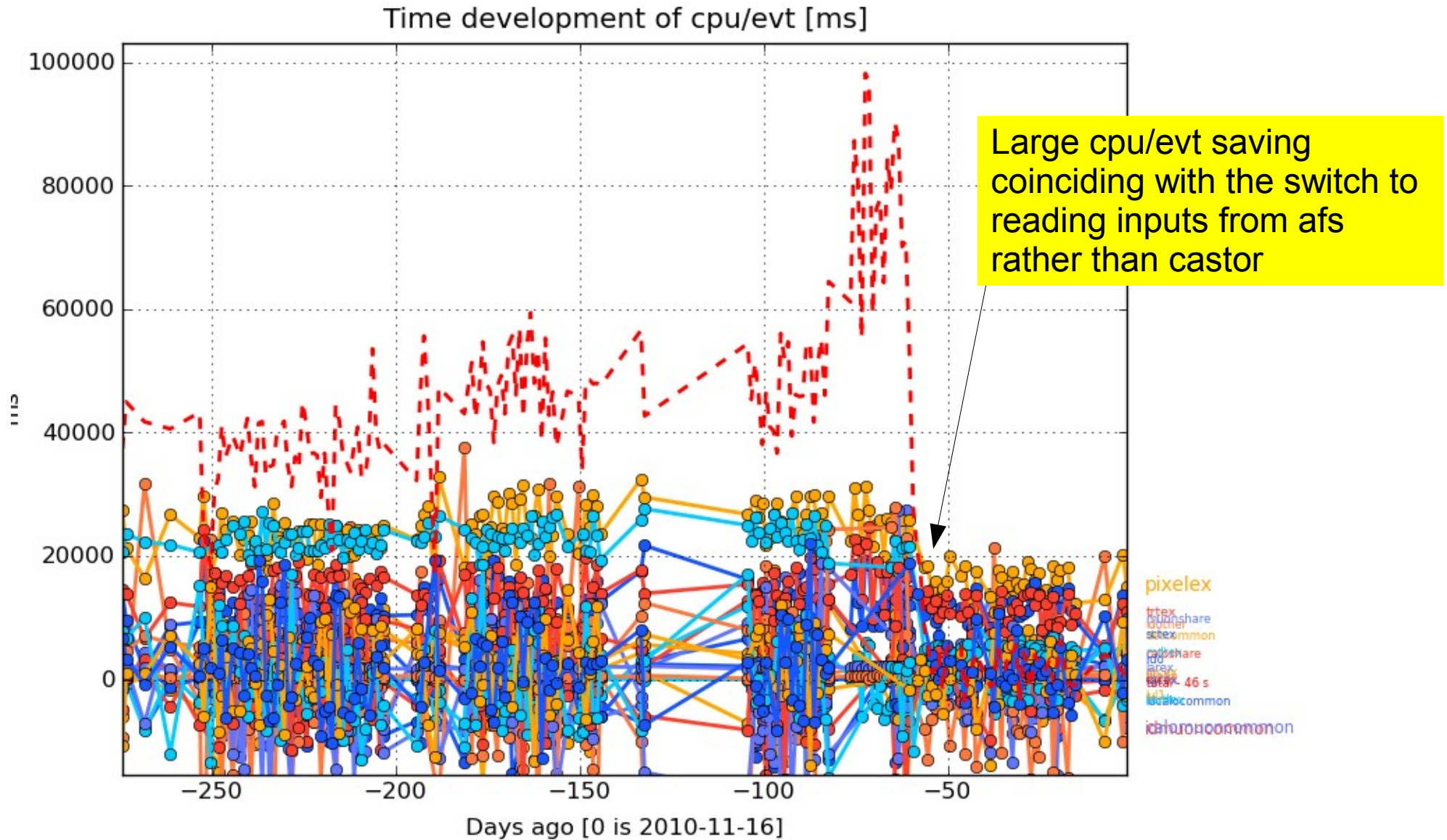
Evt Loop vmem 3.5E33 Pile-up



Evt Loop Malloc Usage 3.5E33 Pile-up



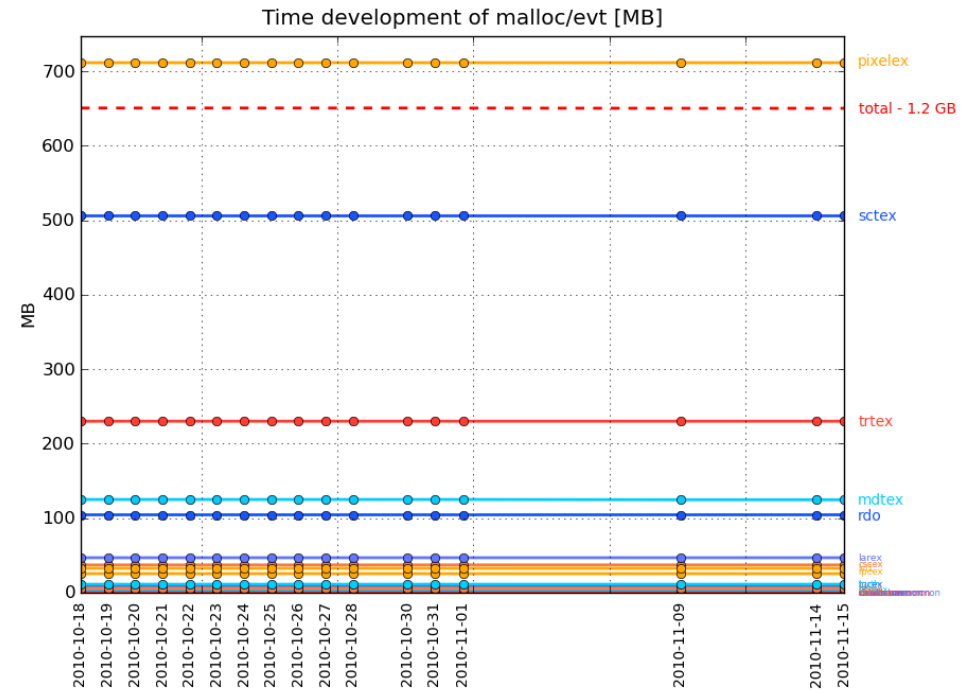
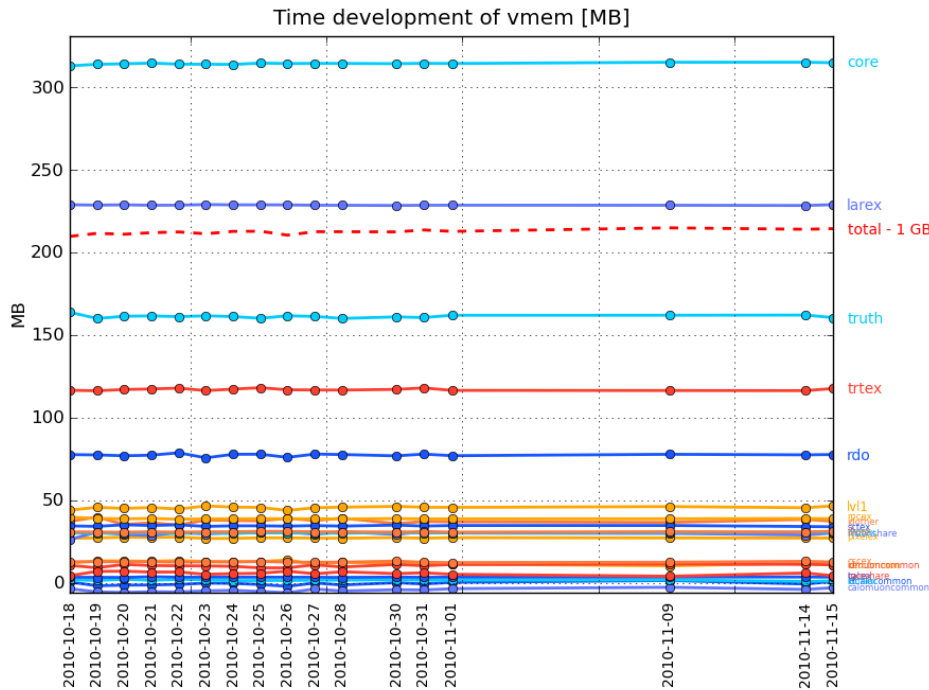
Evt Loop CPU Usage 3.5E33 Pile-up



vmem and malloc Usage - Details

Full information here:

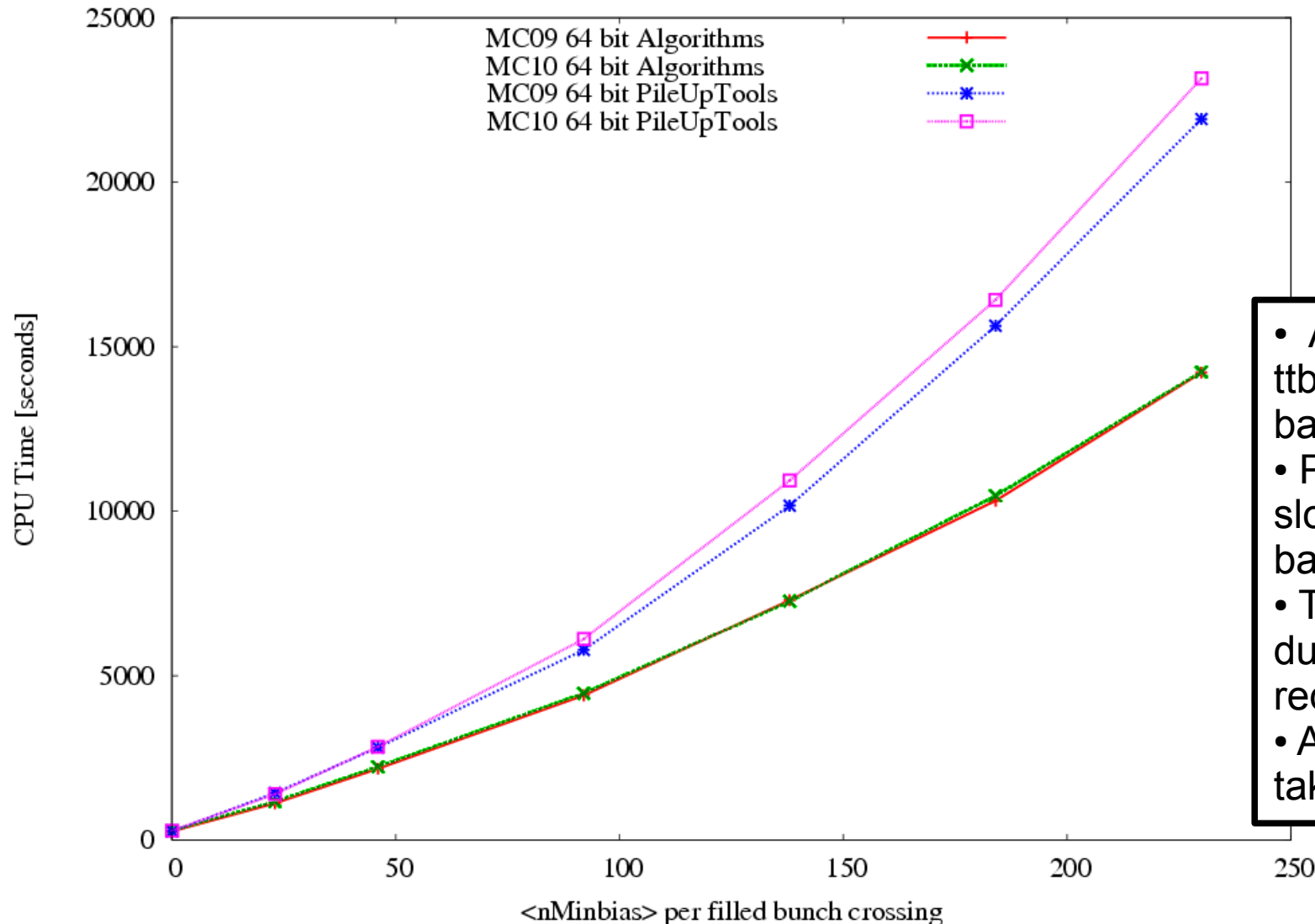
<http://atlaspmc.cern.ch/rtt-mon/?var=vmem&build=dev&cmtconfig=i686-slc5-gcc43-opt&jobset=digipile&cache=prod>



- As before, the above plots are for 3.5E33, 25ns pile-up using mc09 inputs.
- 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 for Pixel to use data pools will be looked into soon.

High Luminosity Pile-up

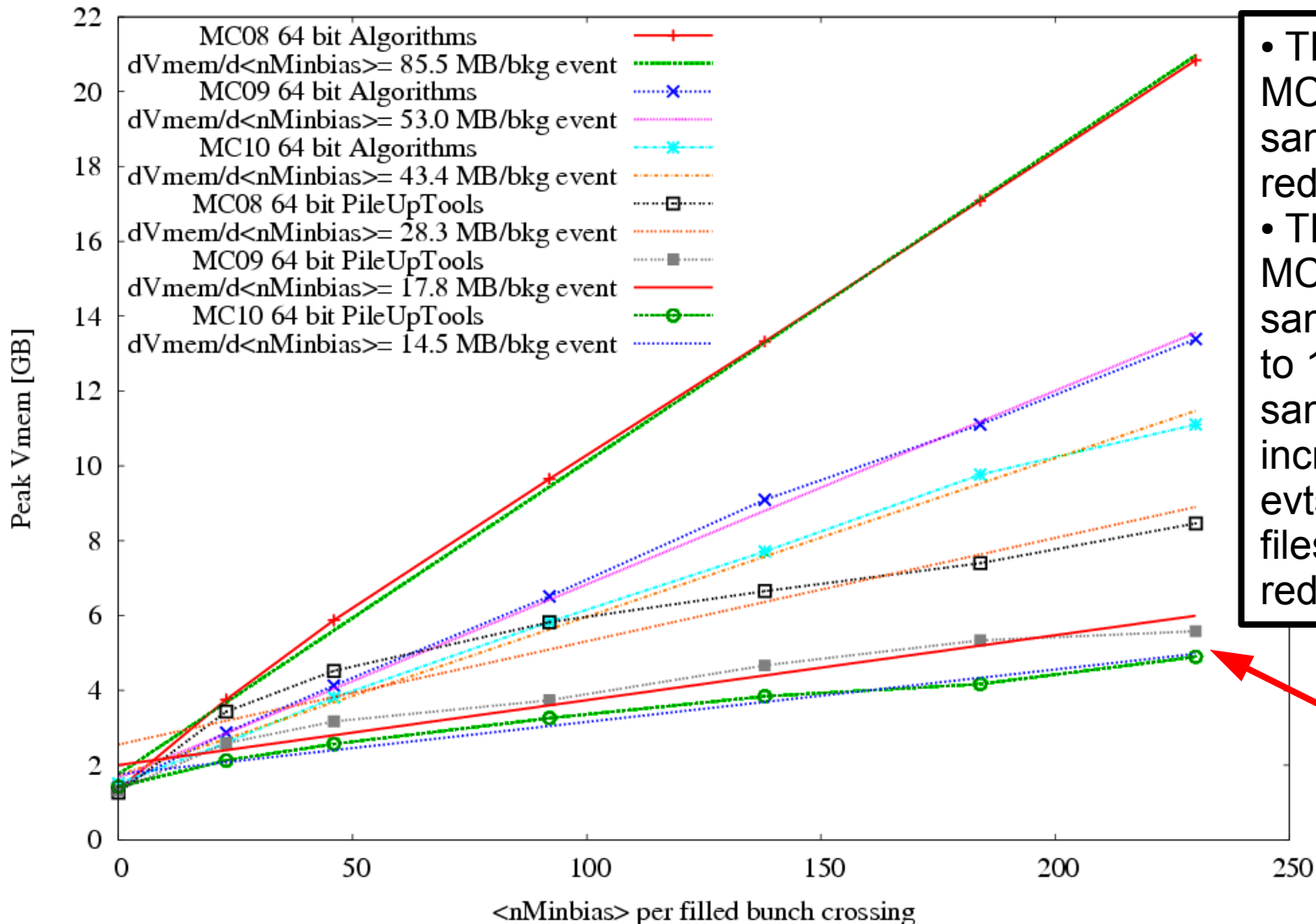
Job Time Scaling with 16.3.0 (50ns bunch spacing)



- 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!

RDO Files Produced by 64-bit Athena

- A few high lumi-pileup RDO files produced with 64-bit athena are available on castor e.g.

```
root://castoratlas//castor/cern.ch/atlas/atlascerngroupdisk/proj-  
sit/digitization/debug/mc10_7TeV.105200.T1_McAtNlo_Jimmy.simul.HITS.e5  
98_s933_Lumi5E34_50ns_PileUp.RDO.pool.root
```

(other luminosities available in the same directory)

- Problems reading them back with 32-bit athena
 - <https://savannah.cern.ch/bugs/?75583>

Necessary Steps Before Production

- Grid queues with higher memory limits ✓
- Validation of 64-bit athena releases.
 - There was a proposal to initially validate 15.6.12.9 for G4Sim and 16.0.2.Y for Digi/Reco
- Rates of Beam Halo/Beam gas interactions to use.
- Validation of PileUpTools
 - Could proceed in parallel to 64-bit validation, but probably only worth doing for 64-bit case.
 - Checking with developers that all recent updates to Algorithms are also in PileUpTools.
- Need to check how Reconstruction copes with high-luminosity.

Future Plans

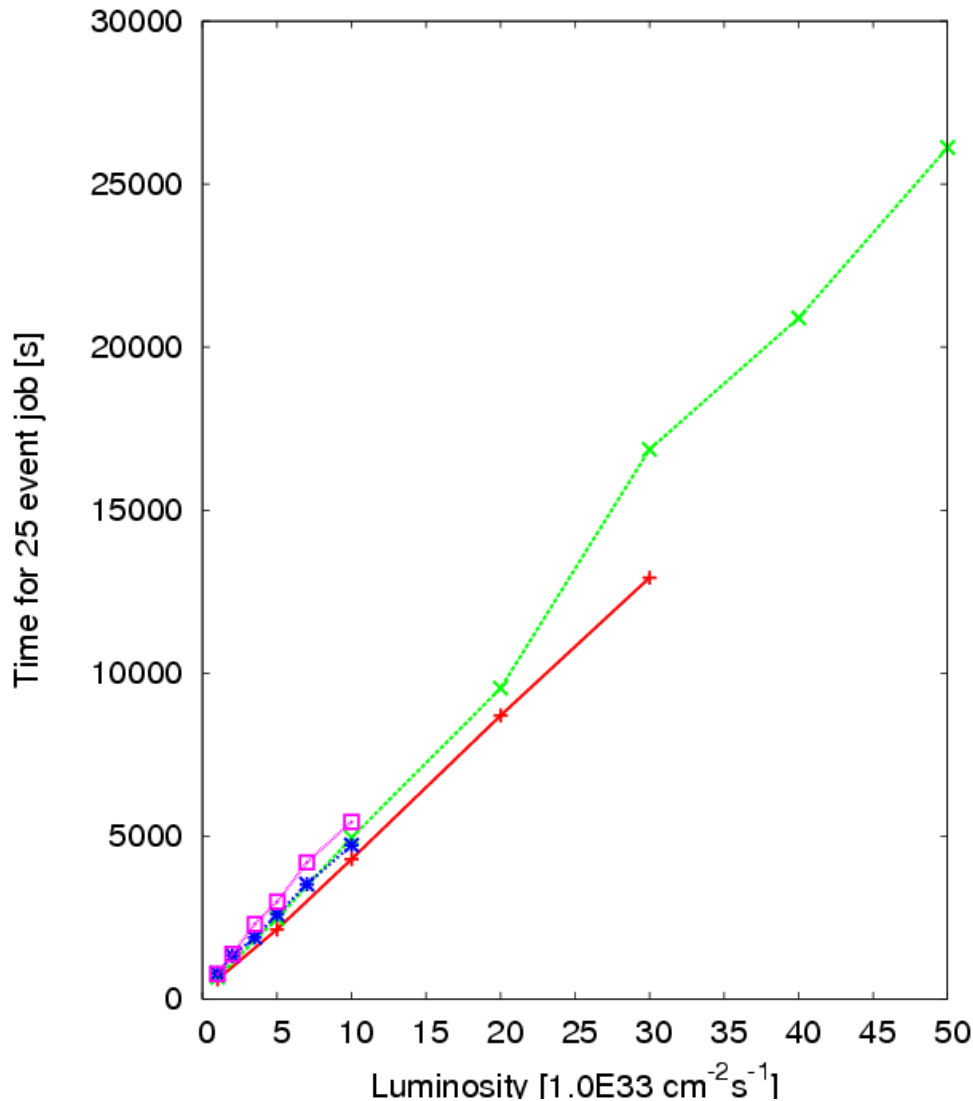
- Continue work on getting PileUpTools validated.
 - Once this is done, work on tidying the code to reduce the maintenance burden on developers.
- Look for further vmem optimizations.
 - Paolo is working on some ideas here already w.r.t closing input files sooner.
 - Take a fresh look at areas for optimization within sub-detector code.

Summary

- Nightly performance tests show reduced resource requirements and more stability since switching to reading inputs from afs rather than castor.
- Big improvement in the vmem usage of high luminosity pile-up jobs!
 - **5E34 pile-up using PileUpTools now requires less than 6GB!**
- PileUpTools ready for validation.
- Will continue to look for further optimizations in vmem and CPU usage.

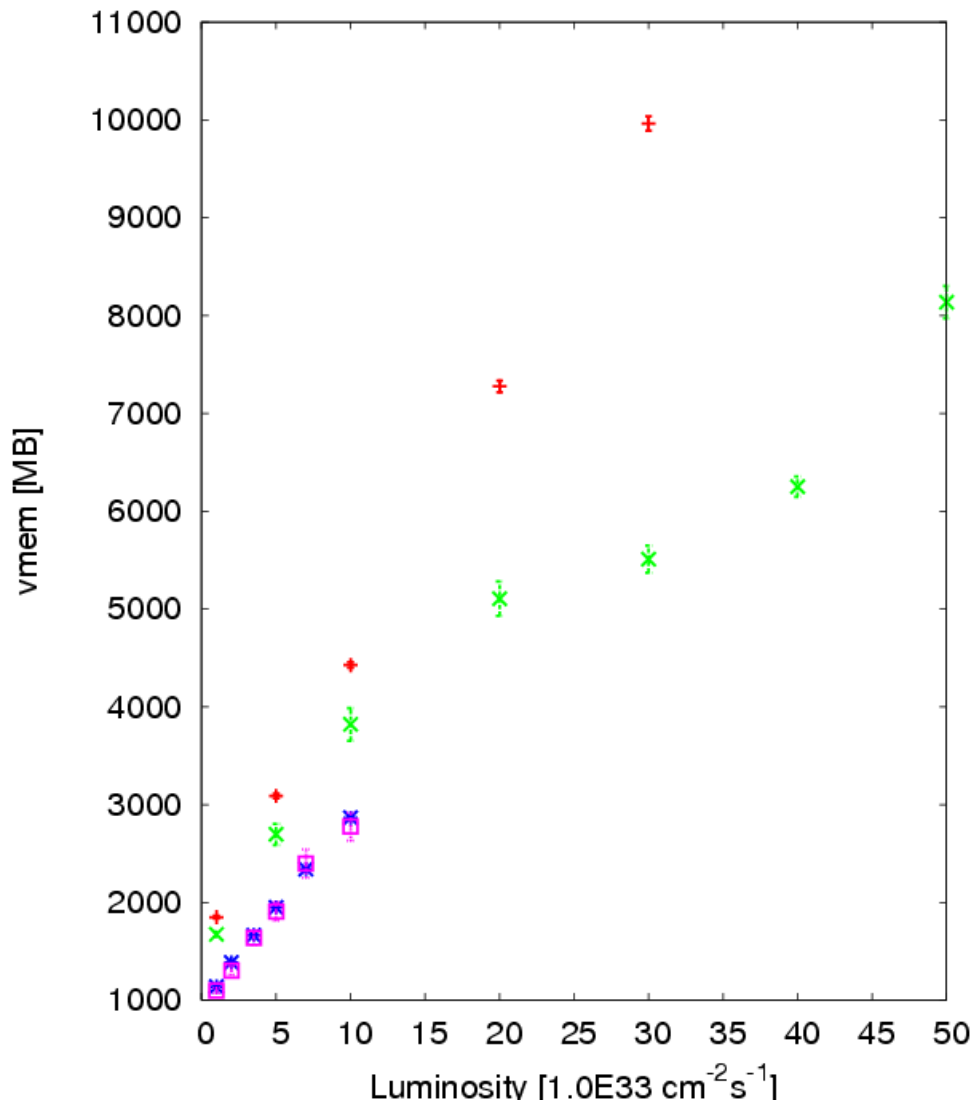
Backup

Job Time Scaling with 15.3.0 (50ns bunch spacing)



- All jobs run over 25 mc08 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 took ~ 7.25 hours.

Memory Scaling with 15.3.0 (50ns bunch spacing)



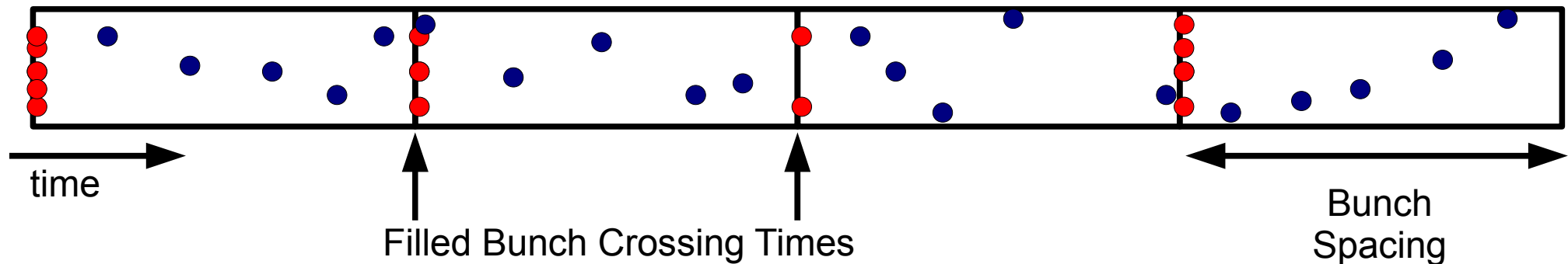
Algorithms 64bit
PileUpTools 64bit
Algorithms 32bit
PileUpTools 32bit

- All jobs run over 25 mc08 ttbar events on the LSF batch system.
- At 3E34 PileUpTools require 4.4GB less memory than Algorithms.
- Possible to run 5E34 pile-up on the LSF using PileUpTools!
- Further reduction in job size should be possible once the problem with I/O overheads is solved.
- 3E34 Algorithm job uses O(1.5GB) more vmem than previous results using rfiio.

Event Timing For Different Background Types

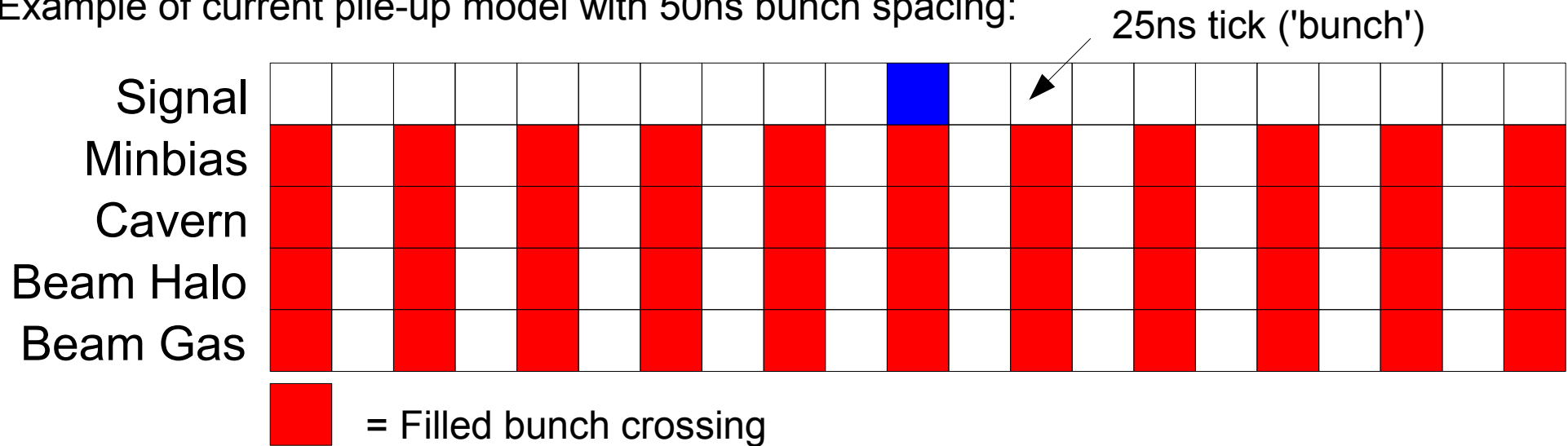
- **Minimum Bias, Beam Halo, Beam Gas:** All events are simulated as starting at the bunch crossing time.
- **Cavern Background:** Separate samples used for each bunch spacing to be simulated. Event starting times are spread over the whole bunch spacing time in order to produce a constant background.

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.

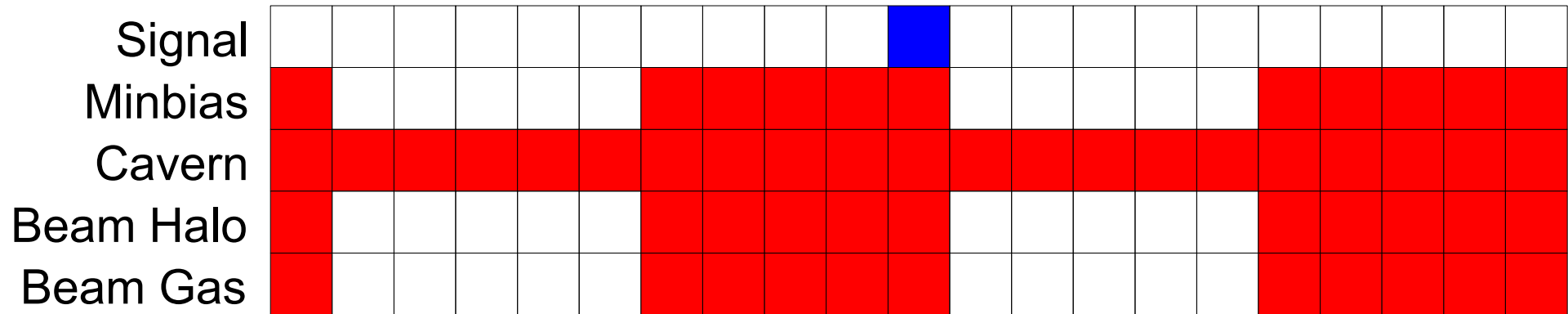


Here there are clear effects on the pile-up and hence detector response depending on where in the bunch train the triggering signal event occurs.

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 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:

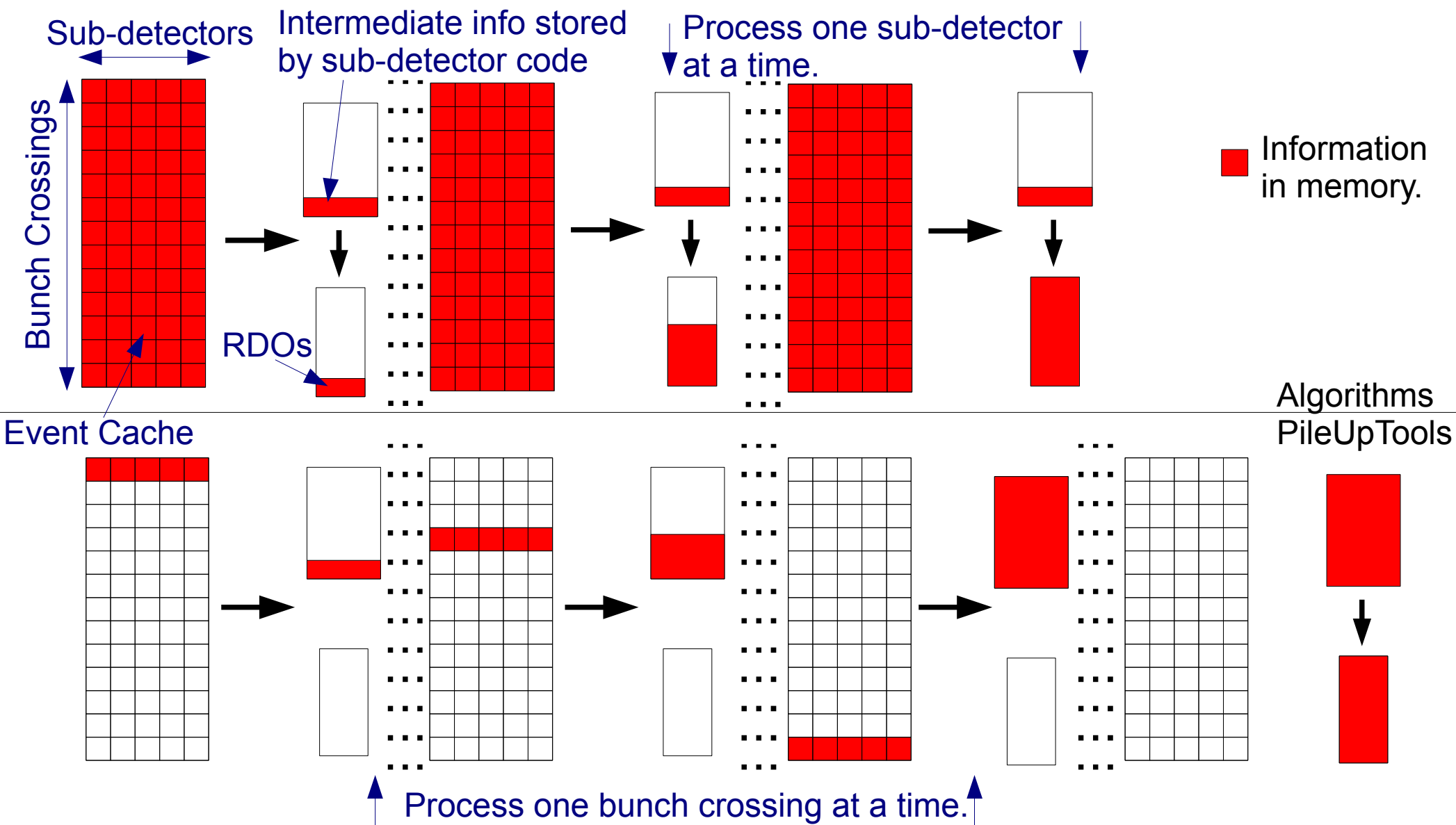
- [FlatBM](#) – reproduces previous pile-up simulation with fixed bunch spacing (default)
- [ArrayBM](#) – signal bunch crossing is picked at random from the filled bunches in the pattern.
- [FixedArrayBM](#) – the same as ArrayBM, but the signal bunch crossing is fixed to a predefined point in the pattern.

<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



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.
- Borut and Pavel working on running new transforms in production with 15.9.0.Y/16.0.0.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.0 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
```

Details of CPU Performance

