Closing remarks

Andrea Giammanco (UCL Louvain-la-Neuve)

Caveats

- Strongly biased selection of topics
- And more than a summary, it's my occasion to babble about my opinions!

Low-hanging fruits?

- Geant4 Collaboration accepts and implement requirements
  - Introduced G4Log and G4Exp
    - Extracted from VDT library (T.Hauth, V.Innocente, D.Piparo)
  - Introduced G4Pow
  - G4PhysicsVector was updated

- For Geant4 10.0 about 5% speed up is achieved by
  - Using fast functions G4Log, G4Exp, G4Pow
  - Optimisation of computation of cross sections

Obviously the Fast MC developers get less pressure than FullSim ones to scratch few % of cpu time here and there, but the same improvements from VDT are expected for us (and for reconstruction), and we are even less in need to justify any small loss of accuracy

Slide from last year

Common tools exist, but you may want to reinvent the wheel anyway

- Delphes is a very popular tool among theorists nowadays (>100 citations; endorsed by LPCC); but use in experimental collaborations (even for future detectors, or upgrades) is limited by preference for a coherent output format between all simulation tools, even if this means reinventing the wheel several times
- But usage of Delphes simulation engine as an external library is possible

What happened in the meantime

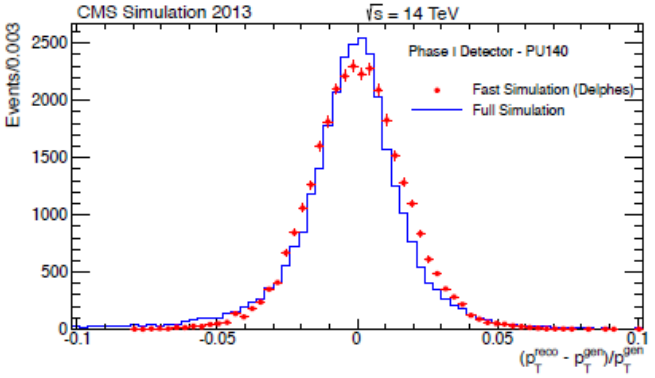
- ATLAS and CMS started using Delphes massively for kick-off upgrade studies
- ...and not as an external library, but standalone
- Possible factors in favour of the change of attitude of the experimentalists towards Delphes:
  - New features in Delphes3, especially pileup related
  - Modularity of Delphes3, which makes adaptations trivial
    - Drawback: tendency to divergence reported, between "CMS Delphes" and "official Delphes" (with some reinvention of the wheel)
  - Human factors: large user base (share ideas); authors answer fast; authors are experimentalists, so they often thought about your experimental problem before you did
- Lesson: maybe, after all, common tools make sense?

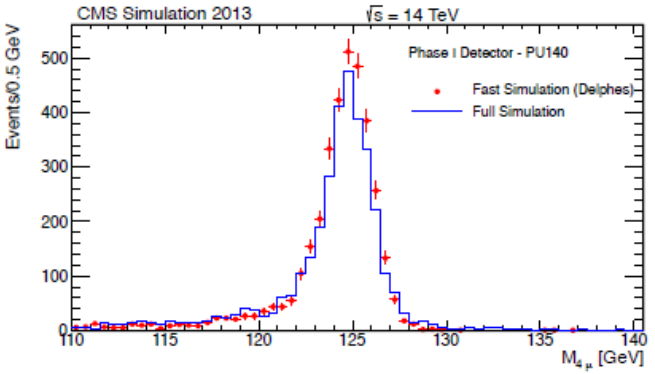
The Delphes dilemma

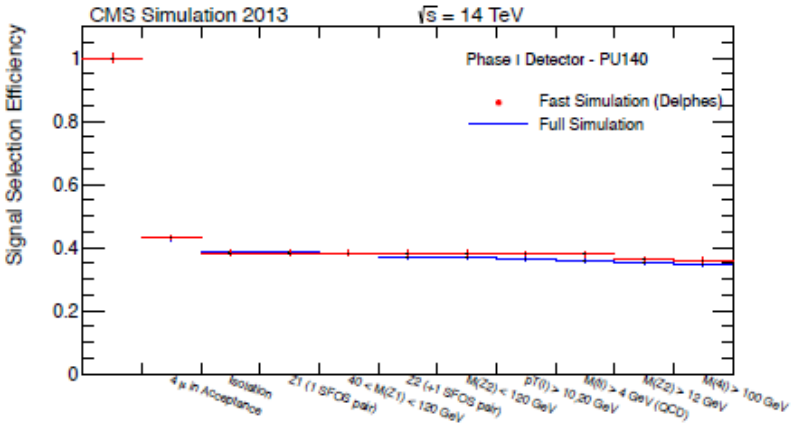
- When should one use Delphes?
- Objection #1: if detector effects are unimportant, one doesn't need Delphes to simulate them
  - You can use Rivet to compare to unfolded results
  - Or apply efficiency parameterization yourself without having to install a new program and write an interface to it
- Objection #2: if detector effects are important, how can such a simple detector simulation be reliable?

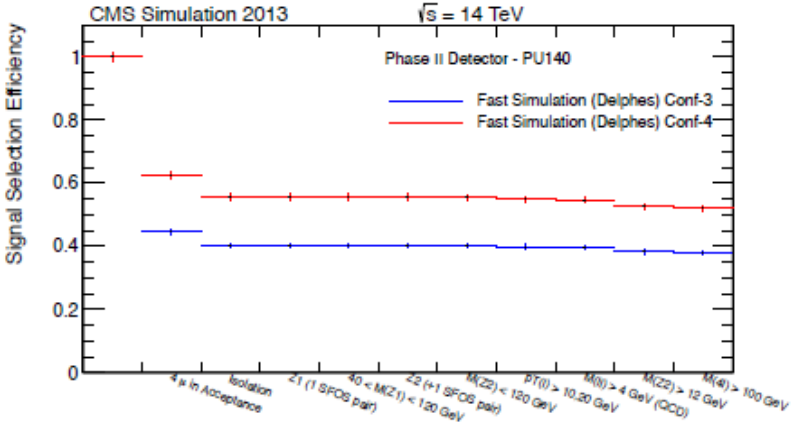
What Delphes Does Well

- Quantities that arise directly from objects with validated parameterization
- H → ZZ → 4μ (CMS-PAS-FTR-13-003)









Objection #1: Delphes works well because at first order what matters is: acceptance, efficiency parametrization and pT smearing; but then why having the Delphes overhead?

Consensus that spontaneously (unconsciously?) emerged: even if any analyst can apply efficiency-and-smearing quickly, cumulative time for N analysts starts to be non-negligible wrt deadlines, and uniformity is guaranteed if all N are using the same tool with the same configuration

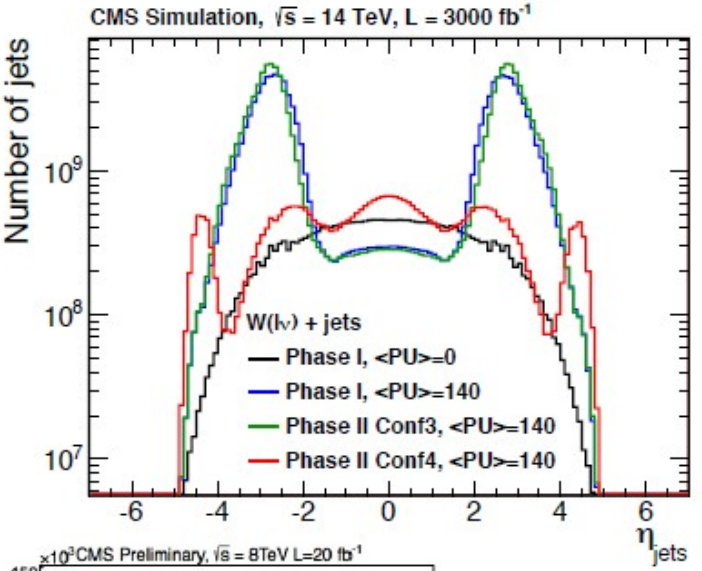
Timing Information

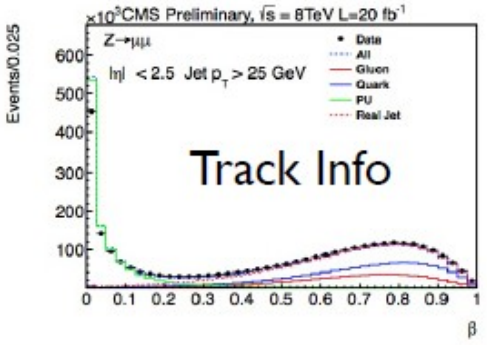
- Use of timing information, e.g. from ECAL cells, is under study for use in pileup rejection after Phase II Upgrade
- Modifications to Delphes made to set vertex timing, extrapolate particle time to calorimeter surface, smear appropriately, and attach information to calorimeter cells and jets
- Very coarse estimate – but this feature is not yet available in full or fast simulation, so this will be the *first* information we have on what timing information can add

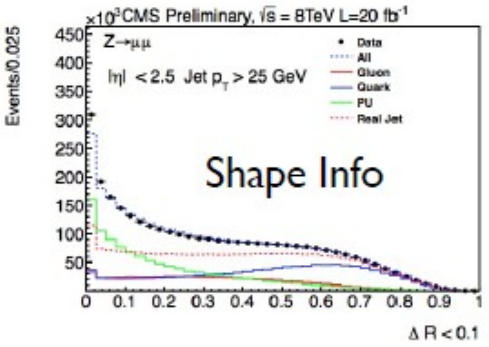
Lesson: people don't start messing with the complex software frameworks of their experiment to add new detector features until they have (quick) indications from parametric studies that the effort is worth being made

PileUp Jet ID

- CMS has a toolkit for identifying pileup-like jets
  - Track information
  - Shape information
  - PAS-JME-13-005
- Now implemented in Delphes

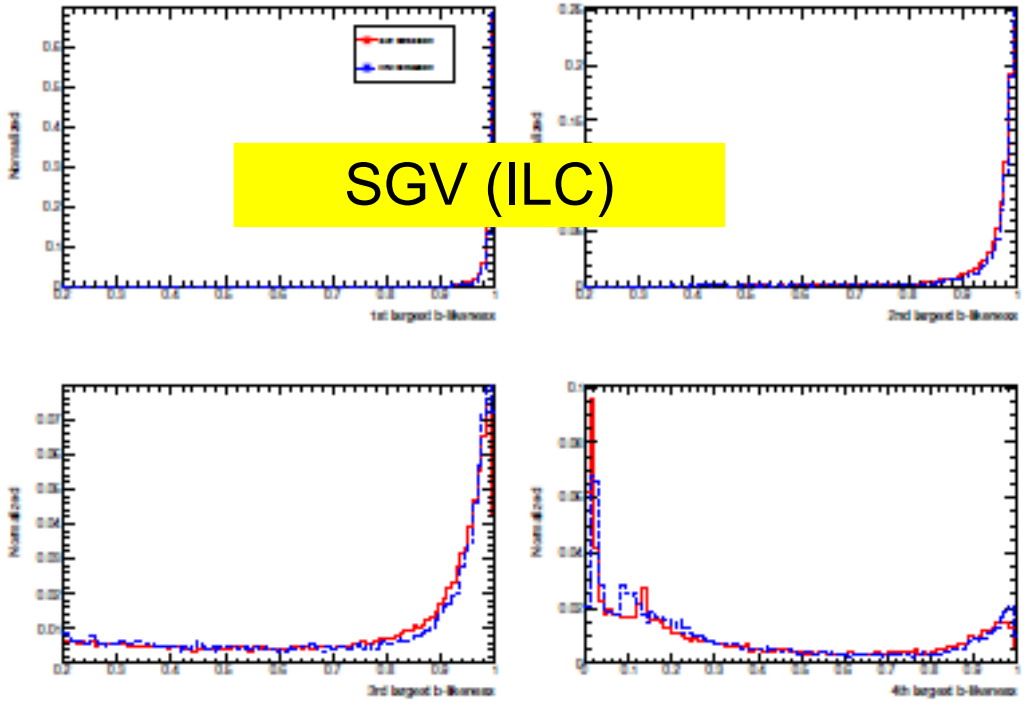






A good example of Delphes as predictive simulation: pileup-induced degradation of jets or MET is not an a-priori input, but it is predicted, and you can study mitigation algorithms before moving to real analysis.

My comment: however, effect on b-tagging cannot be predicted from Delphes, because b-tagging parametrized from parton-level truth; but.....



SGV (ILC)

Running a realistic b-tagging algorithm on top of tracks generated by a parametric simulation gives a fairly decent output... Not %-level accurate, but not completely off

(Nobody says that the extremely complex algorithms of ATLAS/CMS should be reimplemented in Delphes!)

CMS FastSim

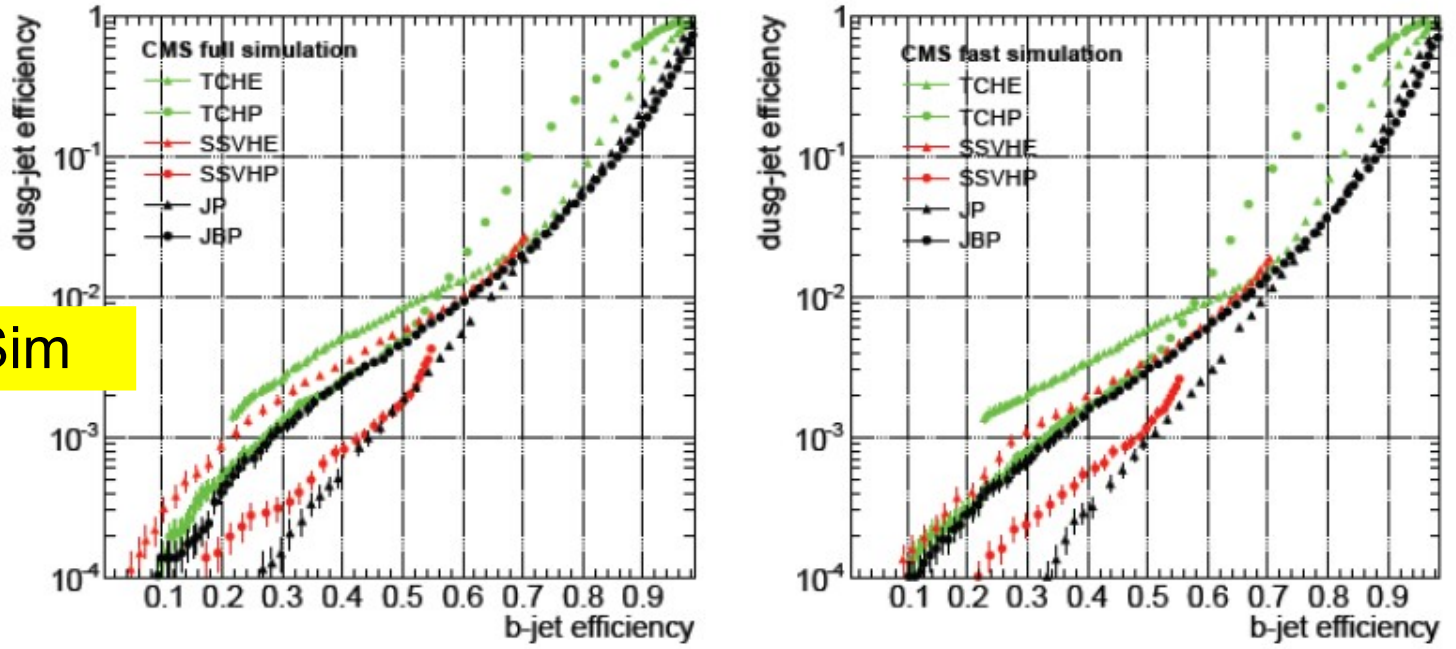


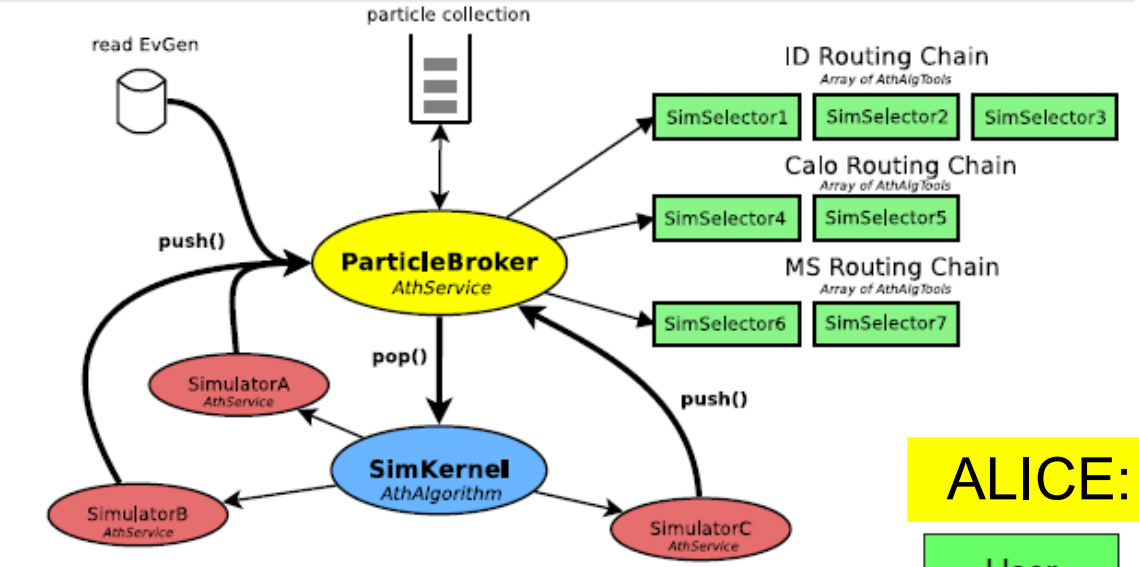
Figure 22: Light flavor mistag efficiency versus b-tagging efficiency in comparison for several b-tagging algorithms. On the left: full simulation, on the right: fast simulation.

Blurring the boundaries between FullSim and FastSim

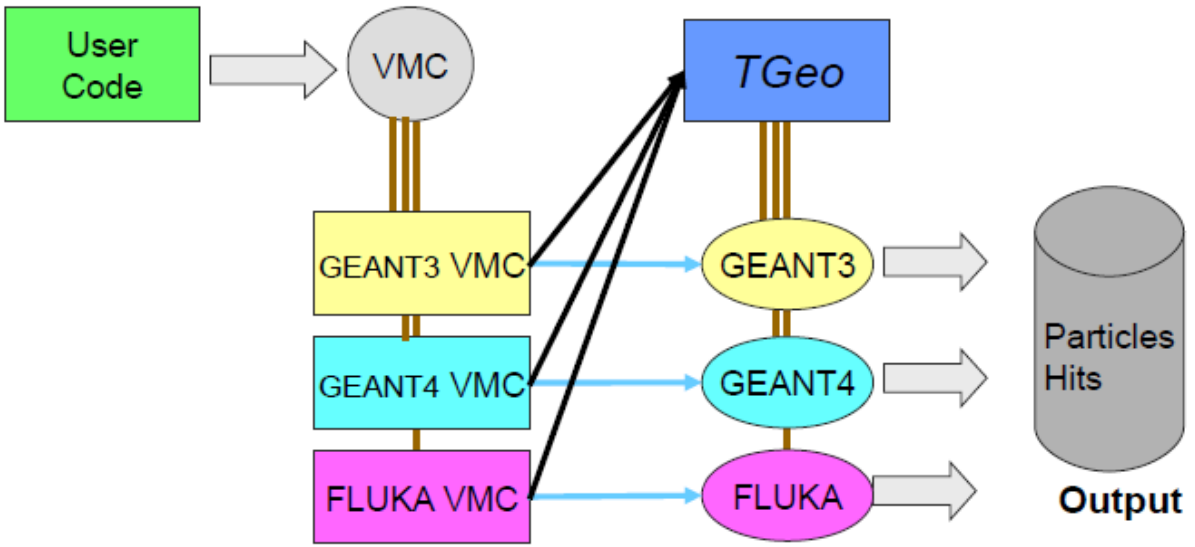
- Several examples of trying to get the best of two worlds:
  - G4 routines called from inside FastSim
  - Pre-simulated data from G4 used from inside FastSim
  - GFlash or Frozen Showers used by FullSim in particularly critical regions (forward calorimeters)
  - FullSim digitizers run in FastSim (and viceversa?)
  - The most extreme example is the ISF approach

Different sim according to particle ID

ATLAS: ISF



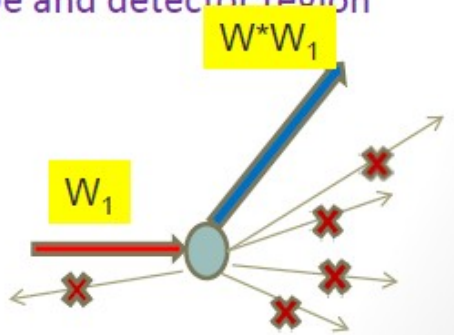
ALICE: VirtualMC



Different sim according to particle ID

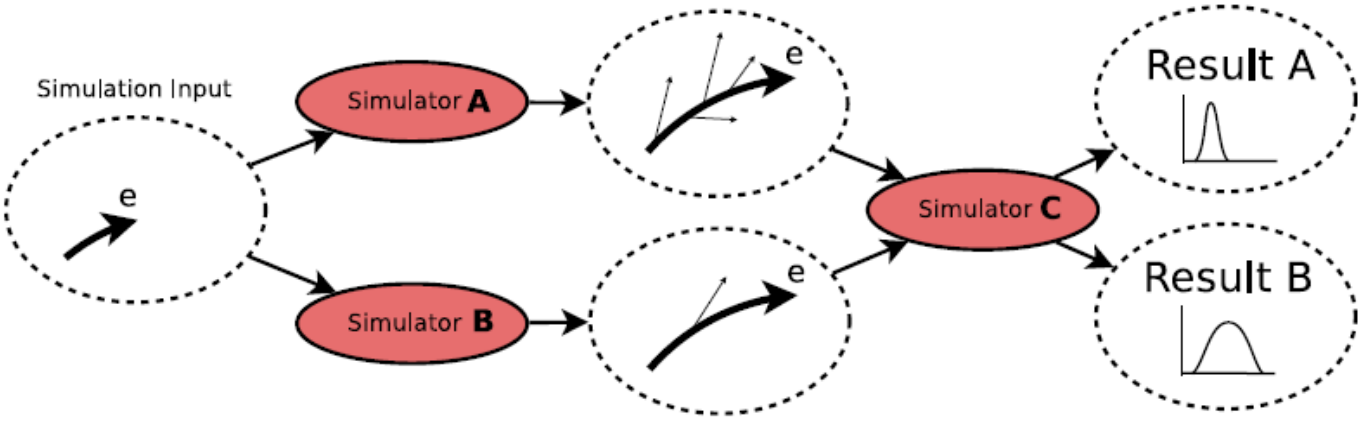
Russian Roulette in CMS

- Method used in neutron shielding calculations for many years
  - Not necessary to track all low-energy particles in a shower
- Some fraction of low-energy particles are killed but remainder get higher weight
  - not suited for tracker, muon systems
  - direct CPU savings (for calorimeter simulation)
  - geometry independent
- RR may be enabled separately per particle type and detector region
  - n, γ - allow significant CPU savings for CMS
  - p, e^- - no visible effect so far
- Two parameters per particle
  - RR factor (1/W)
  - Upper energy limit



New option in G4

Simulator Consistency



Simulator Consistency

- output of *simulator A* and *simulator B* is input for *simulator C*
- simulators A and B have different tunings, energy cuts, ...

→ simulation output for exactly the same generator particle **will be different** between *simulator A* and *simulator B*

⇒ consequently *simulator C* output will be different

→ *simulator C* may need to take into account the originating simulator of a particle

Example Scenario

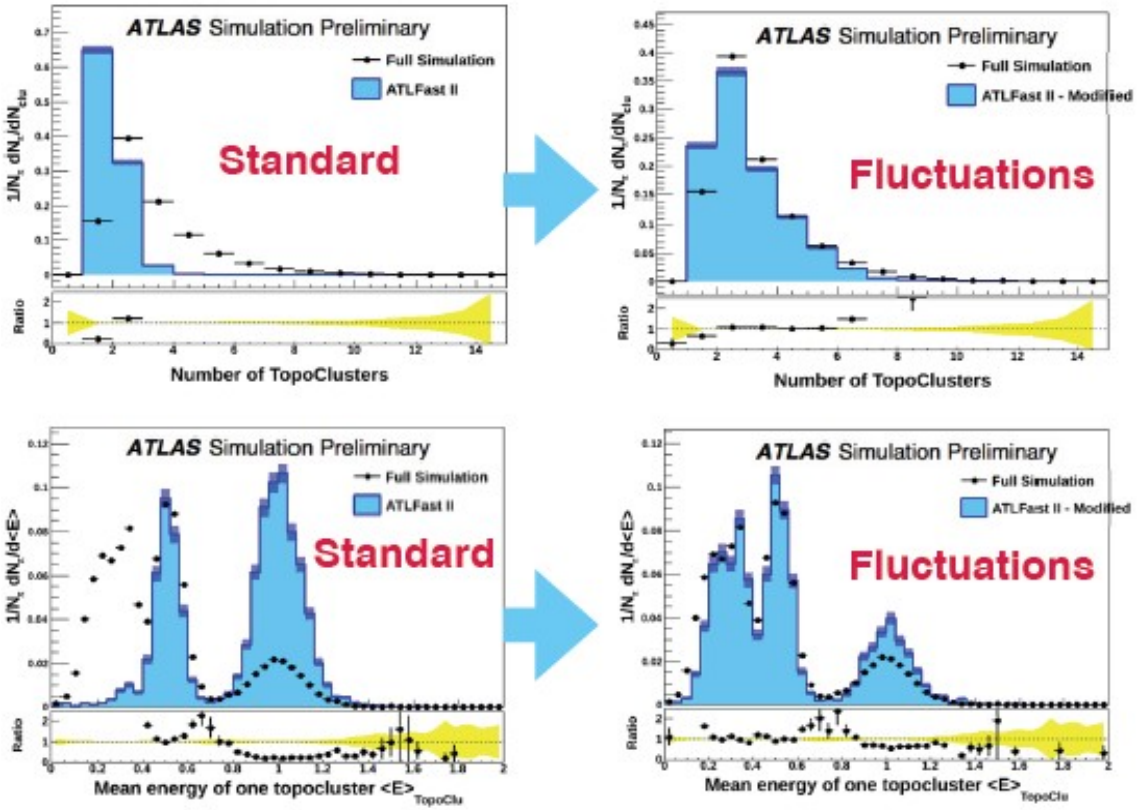
- signal electrons simulated with Geant4 (produces many low-E secondaries)
- rest of ID simulated with Fatras (much higher secondary threshold)
- FastCaloSim takes any of the secondaries for calo simulation
  - → does not distinguish between Fatras or G4 secondaries
  - → will be **over/under-estimating the energy** in the calorimeter

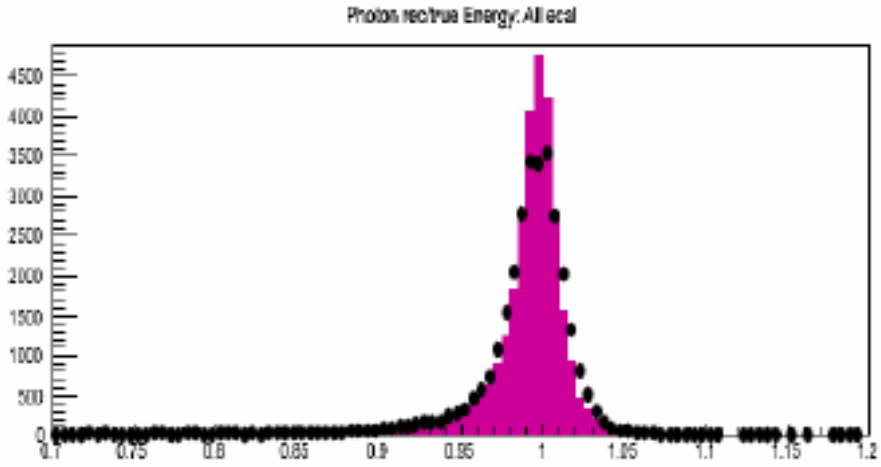
- Embedding of rare signals into background events
  - Saves time of simulation of high-multiplicity background
  - “merging”: generated signal plus generated background
  - “embedding”: generated signal plus measured background

Calibration

- different simulation flavours need different energy calibrations
  - mainly distinguish between full simulation and fast simulation
- what do to when mixing sim flavours (or data) in one event?
  - full sim min bias pileup events and fast sim signal event
  - data zero bias event and full sim signal event
  - different sim flavours within one events (ISF)

Fluctuations

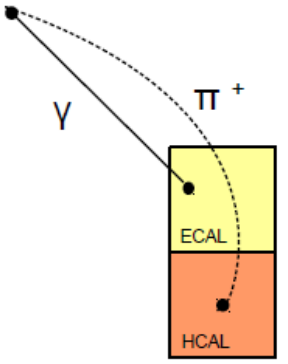




ATLAS FastSim: not enough fluctuations lead to troubles, have to be corrected a posteriori

CMS FullSim with Russian Roulette: too much fluctuation is trouble too – and there is no way to un-fluctuate!

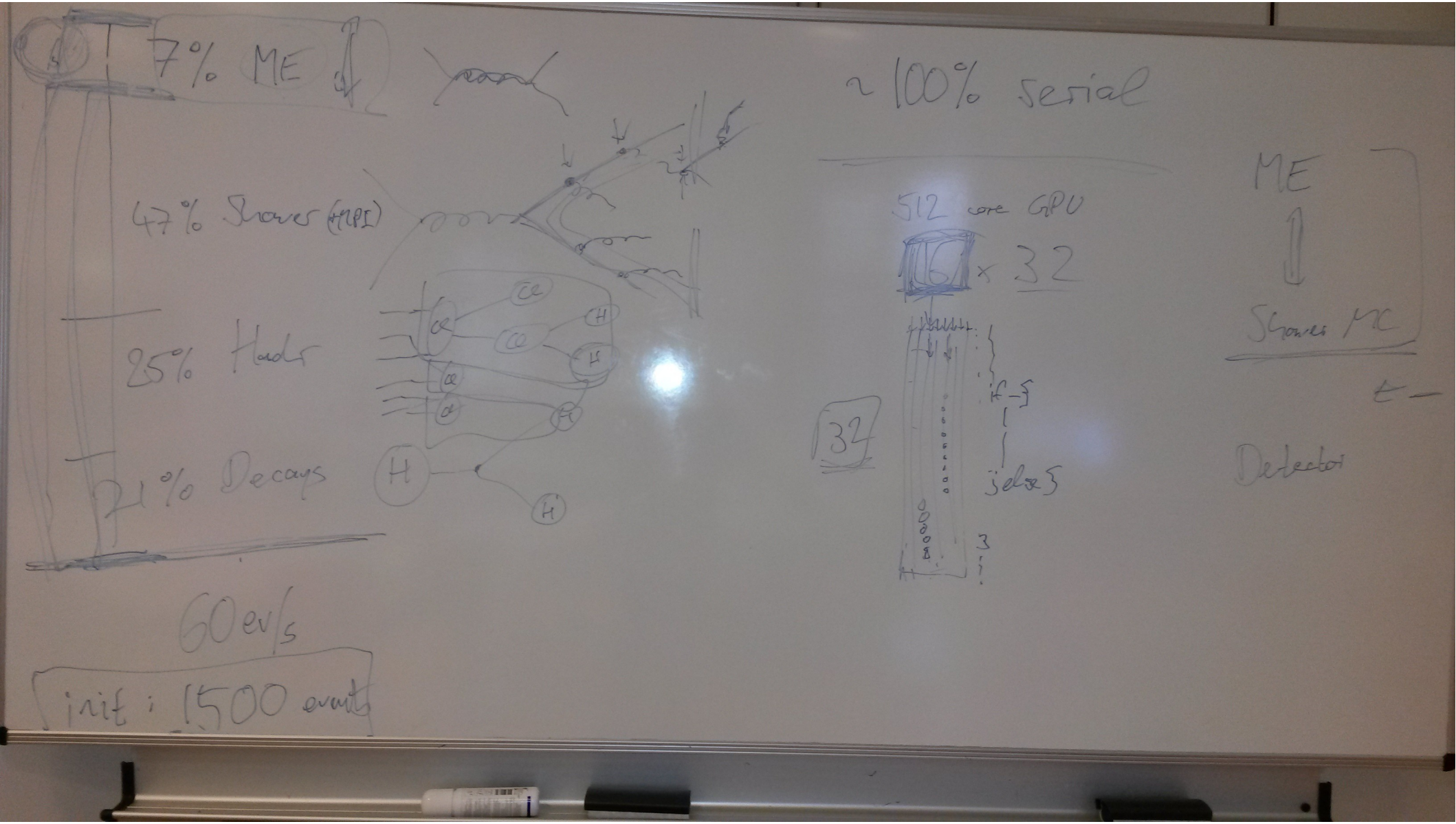
Particle Flow



- Delphes
  - Very crude emulation of PF reconstruction, applying maximum optimism about track-calohit association
  - Decent agreement demonstrated with FullSim across the board
- SGV@ILD
  - They cannot afford to be too crude, as PF is the essence itself of their calorimeter
  - Parameterization of cluster-splitting probabilities seems to work well
  - Any hope to apply a real PF reconstruction on top of simulated clusters?
- CMS FastSim
  - An example of the approach of running the real reconstruction on the simulated low-level objects
  - Some simulation limitations lead to the need for corrections after reco

Generator session

(that I had to skip – here follows Thorsten's summary)



Generator session

- AB:"Traditional Event Generators has been forgotten"
- Tools (Andy) : HepMC; Lhapdf5 → 6: 2 GB → 260kB total memory used (removing static fortran initialization)
- Generators (aMC@NLO, sherpa):
- Integration (to be done once)
  - Clever tricks with virtual terms (reduction of integral and sampling points for different helicity states)
  - Conventional multi-threating saturates at a few CPU O(5): slowest sub-process → M(essage) P(assing) I(nterface) phase integration by many jobs which talk to each other
- Bottle neck is unweighting (=generation) of events:
  - 1000 events/day for complex events
  - Can use GPU for decision and the CPU for events
- In simple event topologies tools (fastjet for clustering → shower matching, lhepdf) get important

THANKS

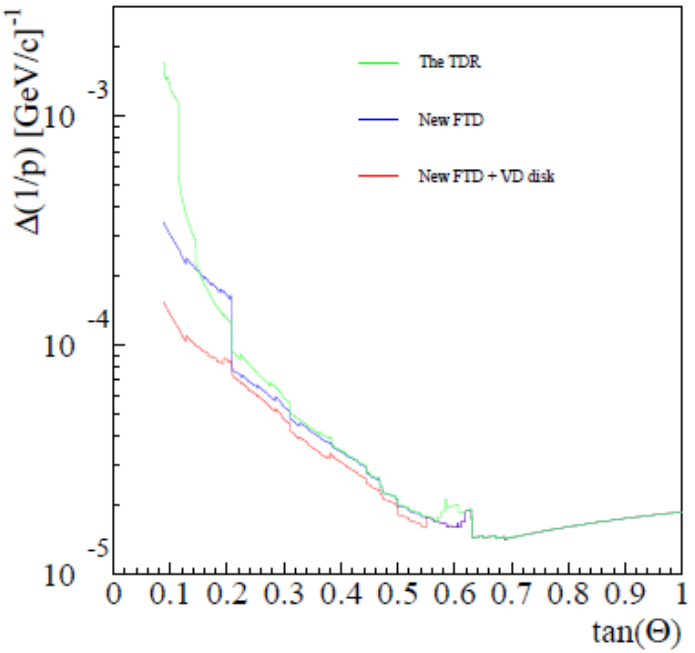
- Thorsten and Andreas as driving forces behind this workshop
- Andreas also for creating the most used infographics of the FastMC community :)
- Martina Mende and Christine Iezzi for organization, webpage, practical help to attendees, etc.
- Manfred Biastoch for typing in all the IP addresses
- Thomas Schoerner-Sadenius, Allianz support
- All of you for coming

Backup

Another example of parametrized simulation that helps detector design

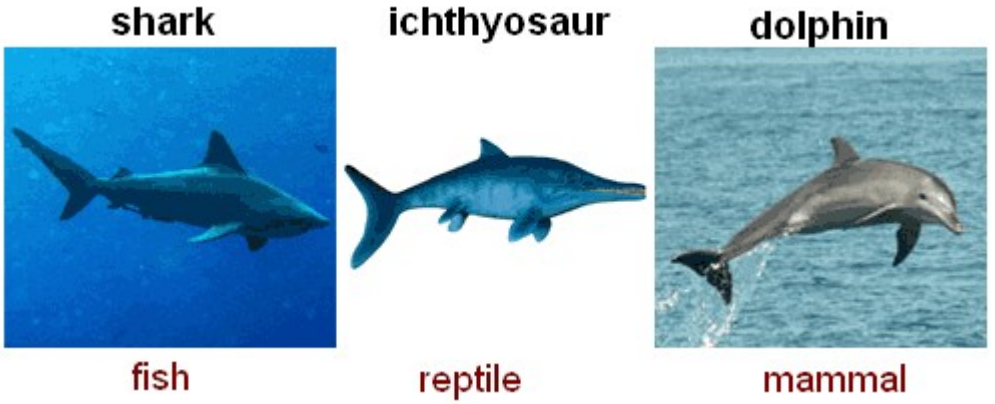
Ex: ILD tracker-system design (Vienna 2005).

- I: The divergence in the TDR: Once the last disk is hit, the $1/\sqrt{\tan\theta}$ is back !
- II: The step: End of The Vertex Detector
- Remedy I:Add *disks* all the way to the end of the TPC (5 more strip-disks)
- Remedy II: Add a *pixel disk* with $\sigma_{point} = 4\mu$ just outside the VD.



Slide from last year

Evolutionary convergence



- Similar solutions reached independently by several groups
- So they are probably good solutions :)
- For example, fast tracking simulations in ATLAS and CMS are very similar both in philosophy and in several implementations
  - What about common libraries (à la Geant) for material effects parameterization?