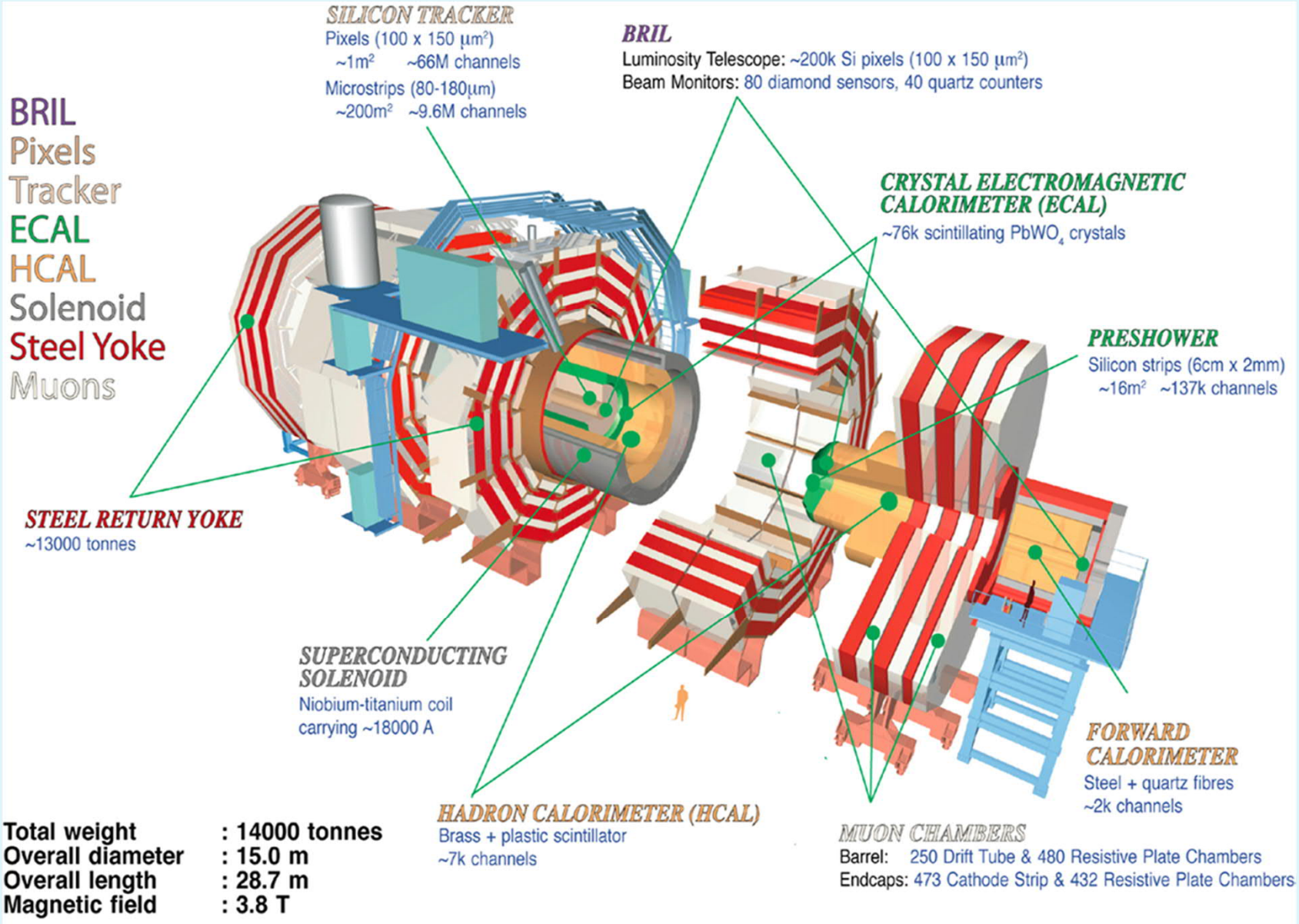
CMS Simulation Software

Vladimir Ivanchenko
Tomsk State University & CERN
MixMax workshop

Outline

- CMS detector
- CMS simulation scheme
- Random engine in CMSSW
- CMSSW validation
- Discussion

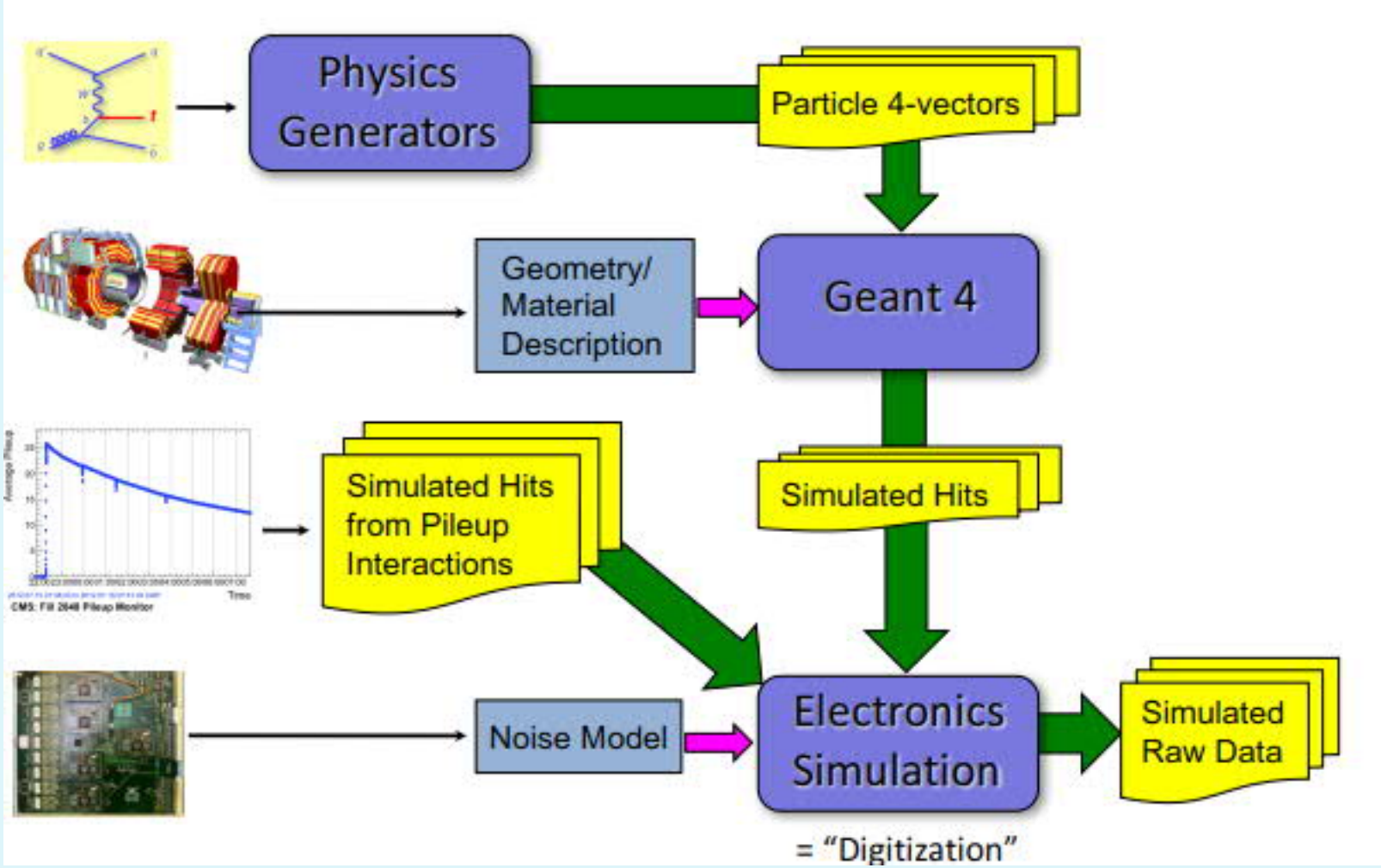
CMS Detector



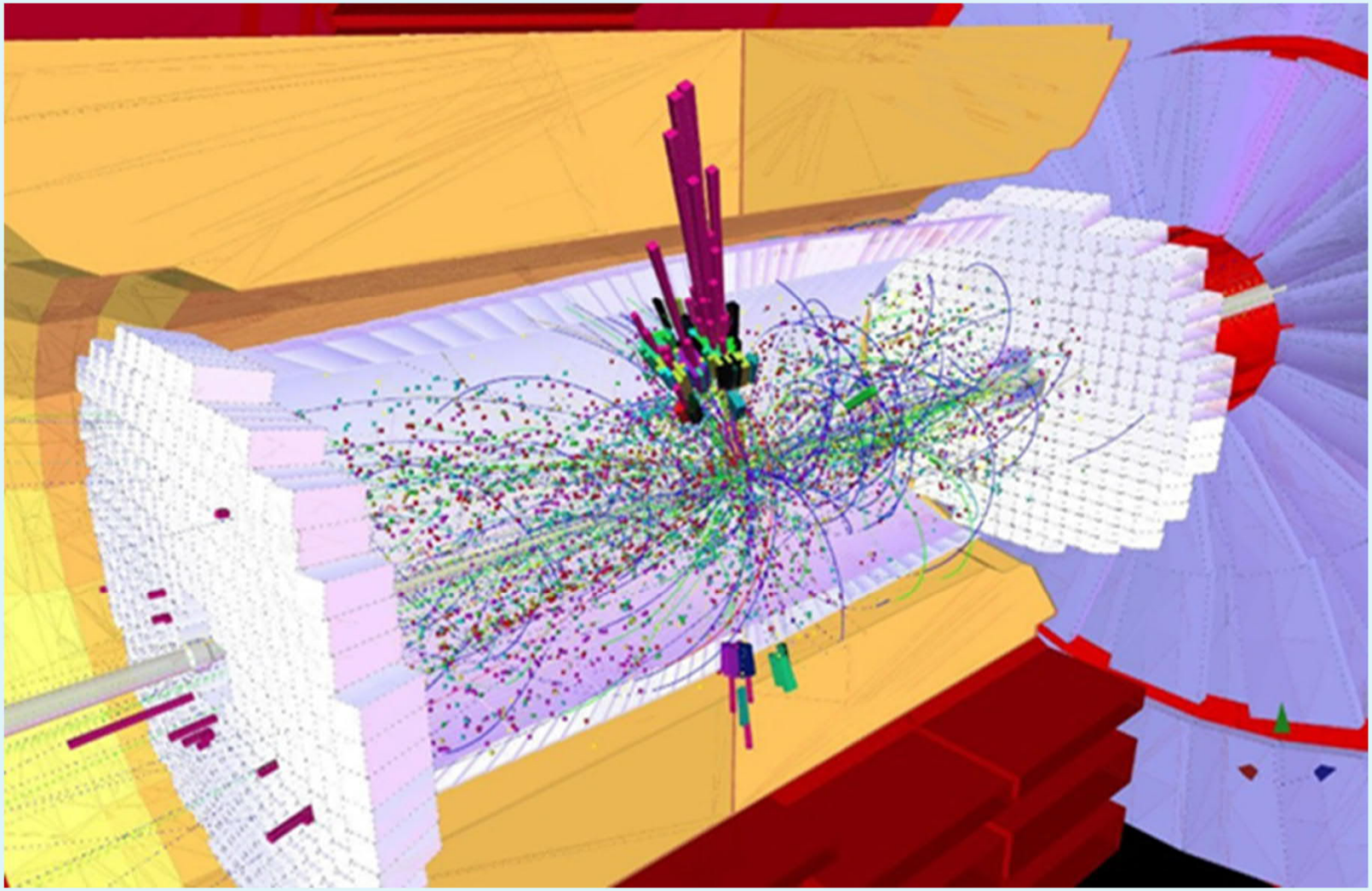
CMS Software

- CMS framework responsible for processing of all type of computation for the CMS experiment
  - Online data acquisition and monitoring
  - High level trigger
  - Simulation
  - Reconstruction
  - Preparation of analysis
- CMSSW framework call execution of Producer
  - Producers are executed one by one
  - Each may use data from previous Producers and produce new data
  - Sequence of Producers may be interrupted and data may be stored into ROOT persistent files
  - Data may be retrieved from the ROOT input file and sequence of producers will be further executed
- Normally, results of Geant4 simulation are stored
  - It is the most time consuming part
- Digitization and reconstruction are performed in a separate run
  - Usually repeted more than once

CMS Monte Carlo Approach



CMS simulated event



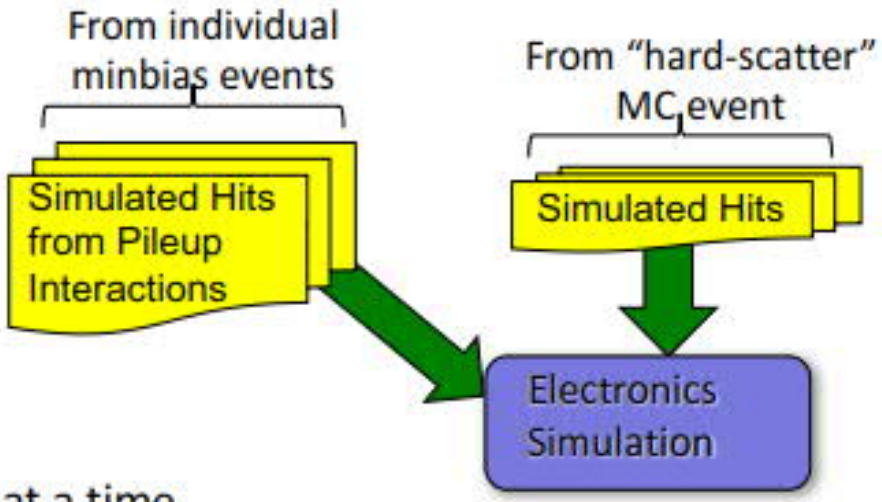
CMS simulation faces significant challenges for both today and tomorrow

- Higher LHC luminosity means:
  - Need for more accurate simulations
  - Need for more events (ideally more events/CHF)
  - More demanding pileup simulation requirements

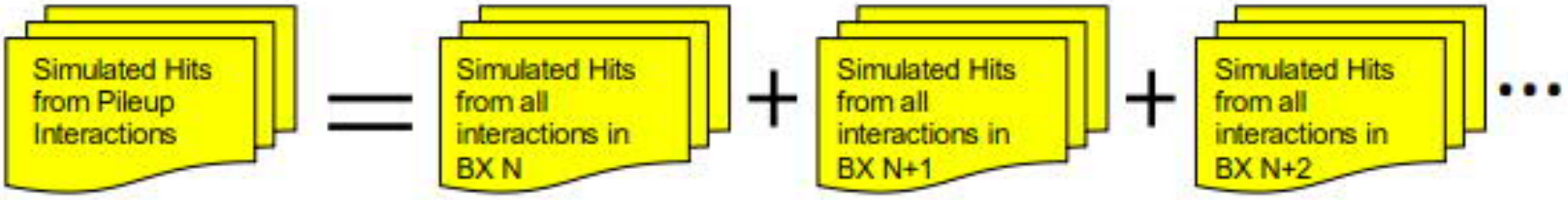
- Major detector upgrades :
  - First was done in 2017
  - Second was for 2018
  - Next will follow during LS-2 and for HL-LHC
    - New detector concepts to develop, benchmark and validate
    - The need to make reliable simulations for HL-LHC luminosities

Simulating extreme luminosities

- Model pileup by including G4hits from MinBias events generated separately from the hard-scatter event
  - Hits are loaded one interaction at a time, processed and accumulated for the final digitized output



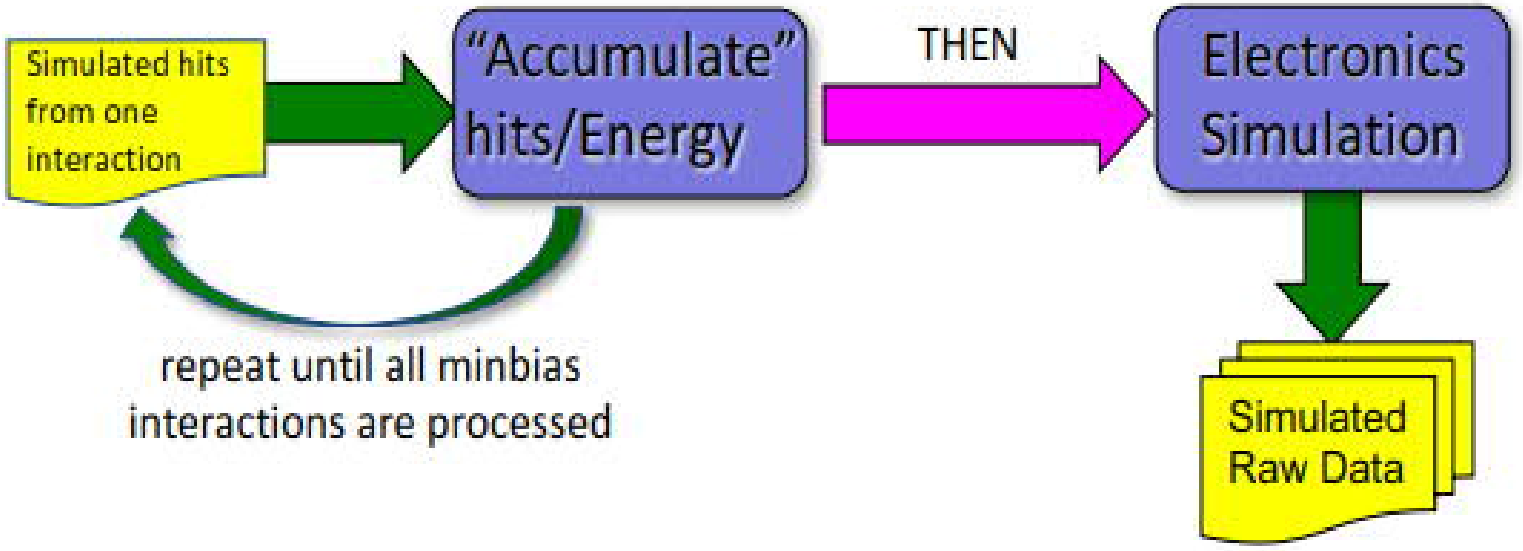
- The pileup interaction simulation is the sum of many interactions



Tens or hundreds of interactions per bunch crossing

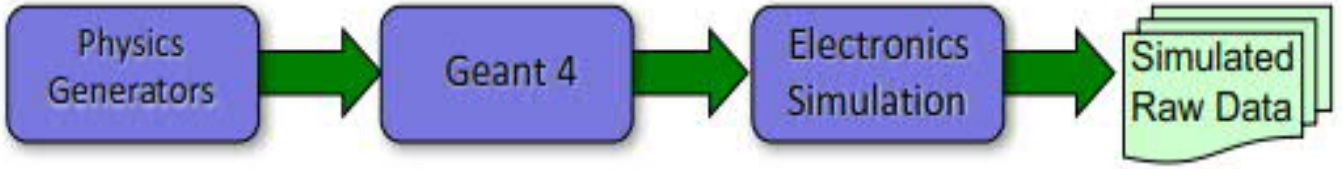
Pre-mixing of pile-up events

- Newly deployed solution: "Pre-Mixing" which proceeds in 2 steps
  1. Upfront I/O intensive step: Create library of events containing only pileup contributions

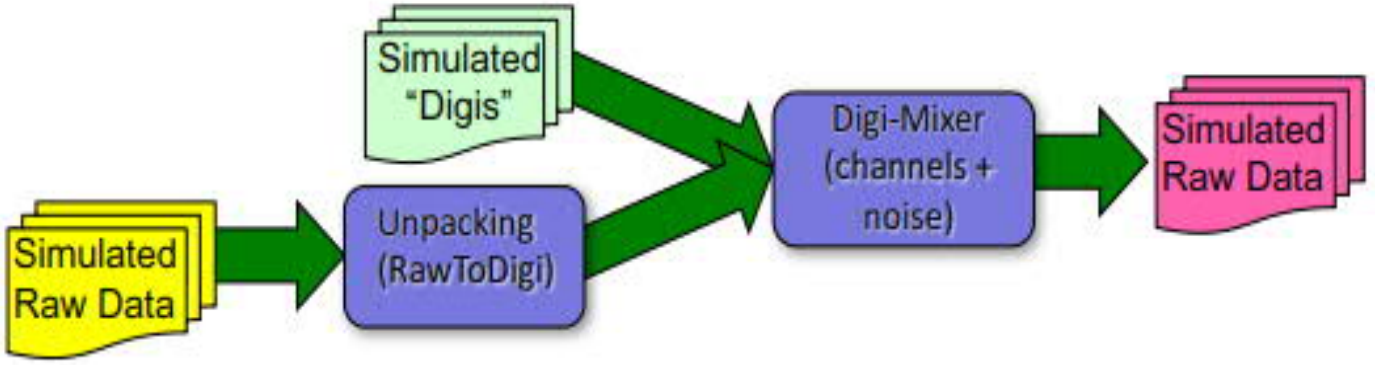


Raw data format at the end of the simulation chain

- The hard-scatter sample is created and processed through the digitization step with no pileup, convert to our raw data format



- Then the two streams are merged



- Only 1 pileup event is needed for each "hard scatter" MC event
  - Much easier to process through computing infrastructure once the premixing sample is created

Premixing lessons learned

- After a long development and validation process, premixing deployed in CMS MC production since 2016

- Issues and benefits we found
  - Extended “raw” format extended to ensure sufficient precision for closely interactions
  - Event reuse: We now potentially re-use entire pileup events instead just individual minimum bias events in Monte Carlo production
  - Flexibility considerations: Generating multiple pileup configurations is now more time consuming
  - Major CPU savings: At current pileup, our digitization+reconstruction processing runs ~2x faster (with a one time cost of the up front production of the premixed library)

Premixing library

- The premixing library is big (~0.1*N(PU) MB/event)
  - Must save sufficient information from the pileup events to allow an accurate digitization
  - This is still a huge savings over the I/O seen in our old approach (~40x). We can run production using remote reads of premixing library
  - Premixing has brought a substantial operational improvement to CMS operations

CMS Simulation production for Run-2

- For 2015-2016 data analysis Geant4 10.0p02 was used and about 18 billion events were produced
- For 2017 data analysis Geant4 10.2p02 was used and about 10 billion events were produced
- 2018 production is not yet started
  - Geant4 10.4 is prepared
  - CMS switched to MixMax

Random number service in CMSSW

- Each CMSSW Producer may call CMSSW Service
  - One of Service classes is RandomeGeneratorService
    - It allows to create random generator engine and initialize it for the Producer
    - Initilisation of the engine is done
      - via configurable parameters
      - Via event record
- In the case of Geant4 simulation the RandomGeneratorService initialize CLHEP random engine

CMS Random number generators

- Random number generators were available:
  - RanecuEngine
  - HepJamesRandom
  - TRandom3
- MixMax was 1st added to the list without real use
  - Fall of 2017
- CMS configure by default
  - For 22 producers the defaults was HepJamesRandom
    - SIM, DIGI
  - For 17 producers the default was TRandom3
    - FastSIM and RECO
  - There are , at least, one place in RECO where std::random is used
- Seeding is done per event

Why transition to MixMax is useful to CMS?

- MixMax demonstrate slightly better CPU performance compared to other generators
  - HepJamesRandom, TRandom3 have nearly the same speed but are a bit slower
- Potentially MixMax allowing remove non-reproducibilities in MT runs
  - MixMaxRng::setSeed(long seed) is reproducible
  - Seed may be defined via some formula depending on run number and event number
- MixMax generator demonstrates the best random properties

How MixMax was proposed to CMS

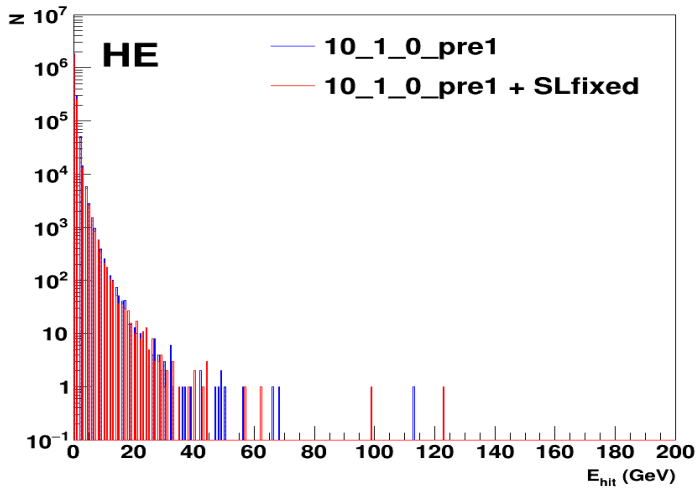
Generator	N random bits	Internal data size	Period log10(q)	TestU01
RanecuEngine	32	430	18	Fails miserably
HepJamesRandom	32	97	43	Fails miserably
TRandom3	32	624	6000	3 tests fail
MixMax	61	38	294	All pass

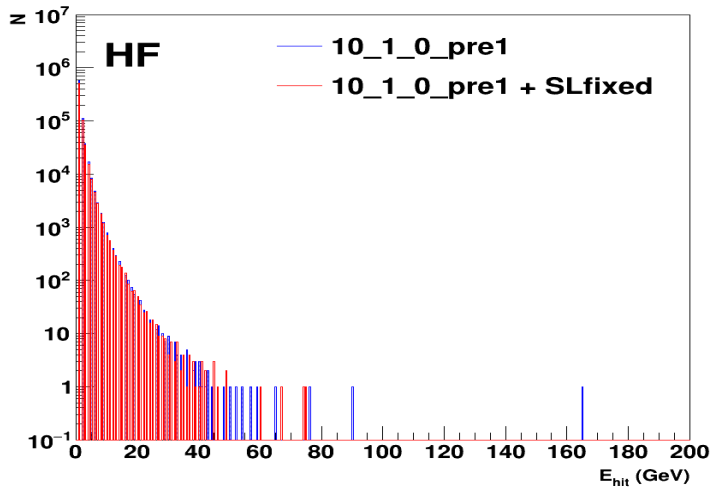
- The table is from G. Savvidy for TestU01: A C Library for Empirical Testing of Random Number Generators, P. L'ECUYER and R. SIMARD (160 tests):
- MixMax obviously have better random properties, which may be essential for HL-HLC
- It was decide adopt MixMax for 2018 massive production

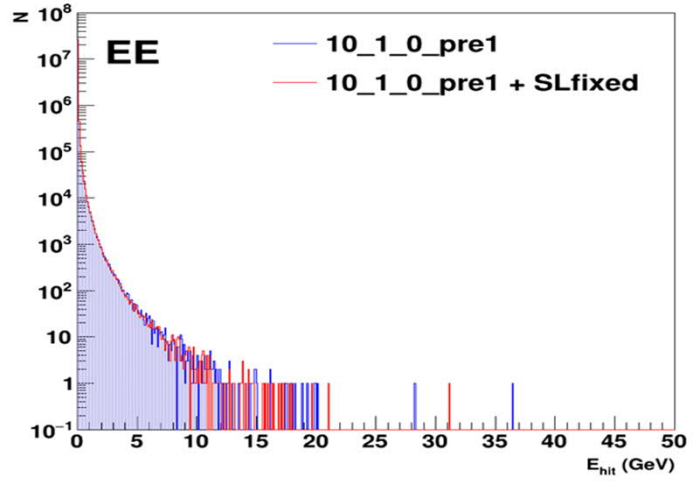
Validation of CMSSW

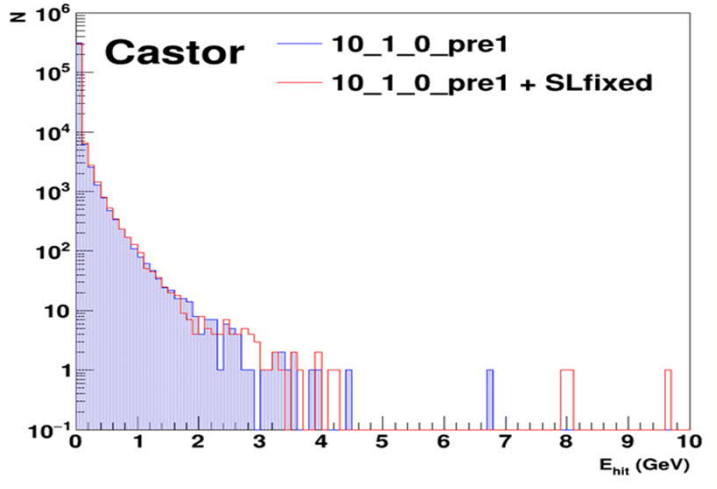
- Any increment of CMSSW is proposed in a form of git pool request, each is tested
  - on code rule via clang code analyzer
  - On compilation
  - On unit tests
  - On set of ~20 WF with regression versus reference version
    - About $2*10^6$ plots used for regression test
    - Statistics is limited to 10 events
- Any reference version is tested with a set of ~20 WF with high statistics 10^4 events
  - Sub-detector and physics analysis groups analyze the results and each sign or reject this reference version
  - If any of $2*10^6$ plot statistically disagree with the previous reference version the explanation of the discrepancy is required

Comparisons of 10k MinBias events









Current status of 2018 simulation

- The CMSSW release is cut just last week
  - MixMax is now the default for all SIM, DIGI FastSim Producers
  - TRandom3 is still used by RECO
- There are some problems in SIM/DIGI which show non-statistical disagreements for muon system
  - Only for the case of pile-up 50
  - Change is beyond statistics when compiler changed from gcc6.3 to gcc7.0
  - Similar differences when we switch from ClassicalRK4 stepper to DormandPrince475
- Main problem remains: how to achieve full reproducibility – results should be the same event if number of threads is different

Discussion

- How validate correctness of usage of random generator engines for huge detector?
- How many pileup events to generate if we know number of desired experiment events and mean pile-up?
- How to handle large fluctuations in a pile-up samples, which may be overlay with different generator events?