Simulation R&D PoW 2020

Witek Pokorski for the Simulation R&D team

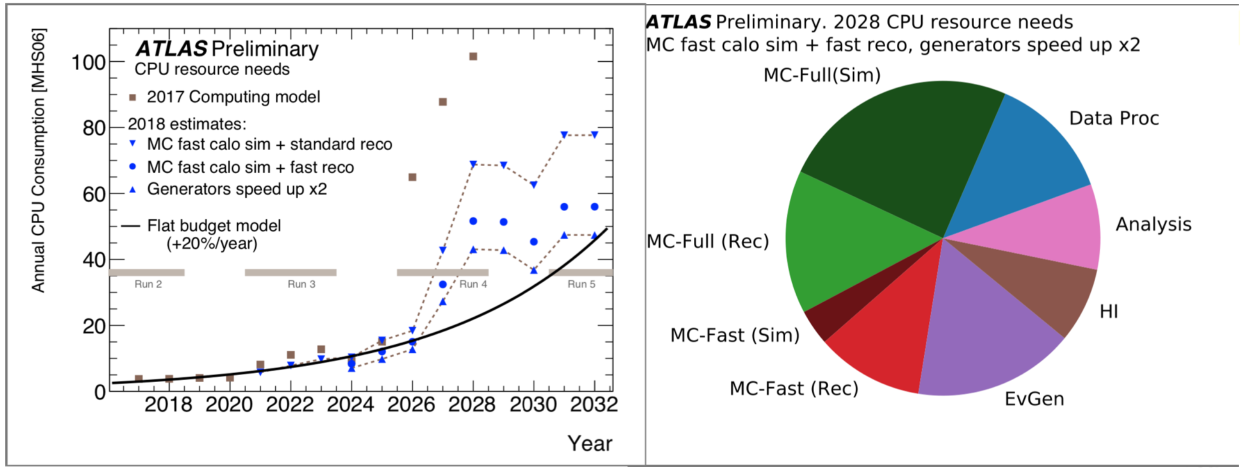
03.02.2020

Motivation - Forecast Simulation Needs

Many physics and performance studies require large datasets of simulated events

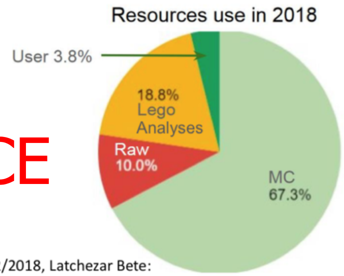
- Geant4 is highly CPU-intensive
- Already lacking statistics -- increasing luminosity poses greater challenges

ATLAS

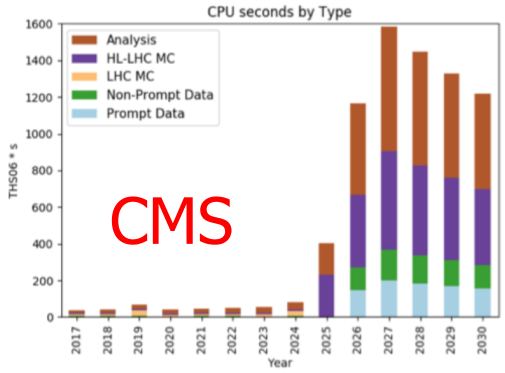


- 2/3 of the computing resources are dedicated to MC simulation, all full sim
  - fast sim not used in production yet
  - fully parametrised fast simulation approach for upgrade studies
- expected 10-100 times more data in Runs 3 and 4
  - cannot cover that with current usage of full sim

ALICE



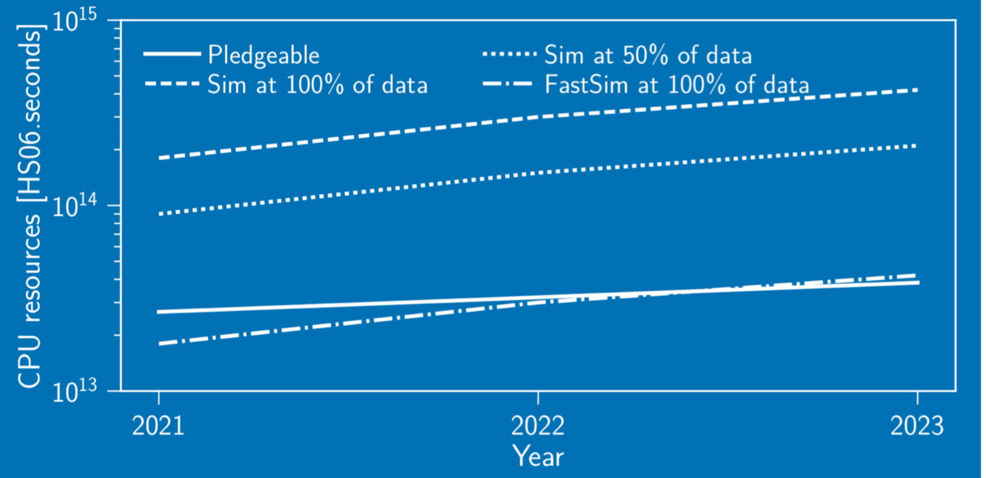
ALICE Week, 12/12/2018, Latchezar Bete:



CMS

- Simulate more events to keep up with HL-LHC data volumes: 10×(Phase1)
- May also need to improve accuracy of physics lists to simulate HGCal
- Reconstruction will take longer due to high pileup and granular detectors

➢ Need more events, more accuracy, in more complicated geometry… w/ relatively smaller fraction of total CPU usage



- Already includes projected optimizations of the software!
- Already today: **very long** waiting-times for samples!

LHCb

Approach - Three Main Axes of Development

- **Improve, optimise and modernise** the existing Geant4 code to gain in performance and precision for the detailed simulation

- Trade precision for performance using **fast simulation techniques** both with parameterisations and with ML methods, and integrate them seamlessly in Geant4

- Investigate the **use of ‘accelerators’** such as GPUs for performance gain

Performance: main directions

GEANT4
A SIMULATION TOOLKIT

Parallelism

Concurrency model review - fine grain parallelism

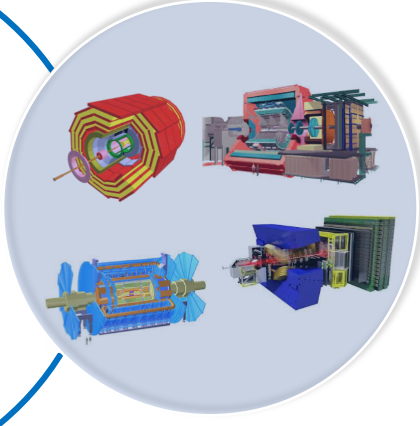
Optimization

Faster physics/geometry algorithms - low level code optimizations

Restructuring

More compact code & data - simplified calling sequence - stateless - pipelines for heavy computation kernels

Experiments integration



Heterogenous computing

GPU friendly kernels

Fast sim revisiting

Parameterizations - ML

(CERN) Team

- Andrei
- John
- Guilherme

- Anna
- Ioana

- Witek

- Mihaly
- Alberto
- Gabriele

(+ 2 summer ‘19 students)

Activities in 2019

- GeantV Vector prototype
  - to assess the achievable speedup using a novel, vectorized approach to particle transport
    - demonstrator of full EM shower in realistic (CMS) geometry and comparison to Geant4
- Fast simulation
  - revisiting classical parameterization
  - novel Machine-Learning based fast simulation R&D

GeantV in 2019

Outcome of GeantV prototype

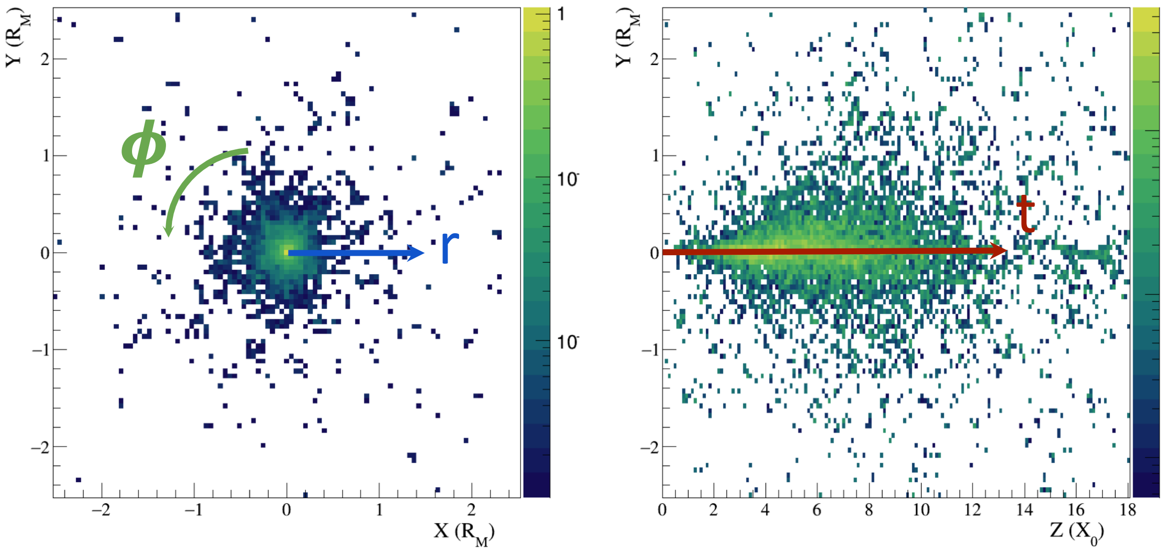
- HSF GeantV dedicated meeting (15 Oct 2019)
- paper reporting all the results in preparation and will be sent for publication soon
- new tools delivered to the community
    - VecCore, VecGeom, VecMath
    - successfully integrated into Geant4, ROOT, etc
- better understanding of possibility of ’vectorization’ of the simulation code

Main lessons

- Main factors in the speedup seem to include
  - Better cache use (single track mode shows performance decrease for small caches)
  - Tighter code (e.g., less classes, indirections and branching)
- Vectorization's impact (much) smaller than hoped for
  - Small fraction of the code has been vectorized or is run in vector mode effectively
  - Overhead of basketization cost similar to vector gain for "small" modules
  - Basketization can bring benefits for FP hotspots (e.g. magnetic field, multiple scattering)
- Basketization cost in
  - Either extra memory copy (using collection of tracks)
  - Or lower memory access coherency (using collection of pointers)

Fast Simulation in 2019

'Classical' EM shower parametrisation



$$dE(\bar{r}) = E f(t)\, dr\, f(\phi)\, d\phi\, dt\, f(r)$$

$$f(t) = \frac{(\alpha-1)^{\alpha}\, t^{\alpha-1}\, e^{-(\alpha-1)t/T}}{T^{\alpha}\Gamma(\alpha)}$$

$$f(\phi) = \frac{1}{2\pi}$$

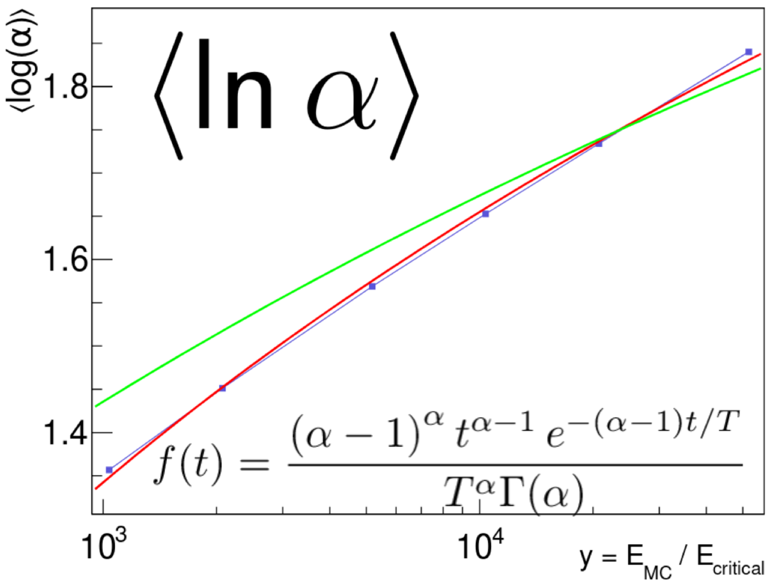
$$f(r) = p(t)\frac{2rR_C(t)^2}{(r^2+R_C(t)^2)^2} + (1-p(t))\,\frac{2rR_T(t)^2}{(r^2+R_T(t)^2)^2}$$

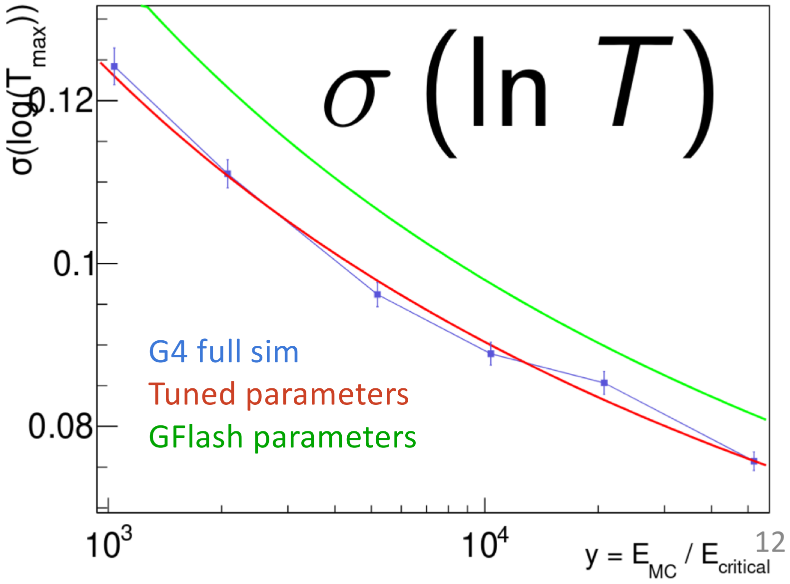
- Based on GFlash model (arXiv:hep-ex/0001020)
- Basic implementation in Geant4:
  - Only for homogeneous media
  - Hard-coded parameters from paper (and GEANT3)

‘Classical’ EM shower parametrisation

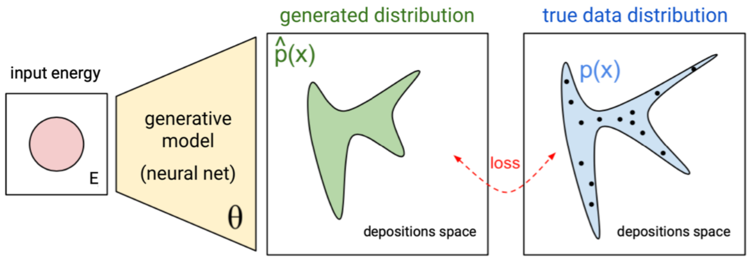
What has been done in 2019:

- ➔ Review of existing code
- ➔ Bug fixes (parallel world & processes in G4)
- ➔ Addition of new examples & validation tests
- ➔ Work on tuning procedure for parameters
  - ◆ longitudinal profile ready
  - ◆ transverse profile on-going

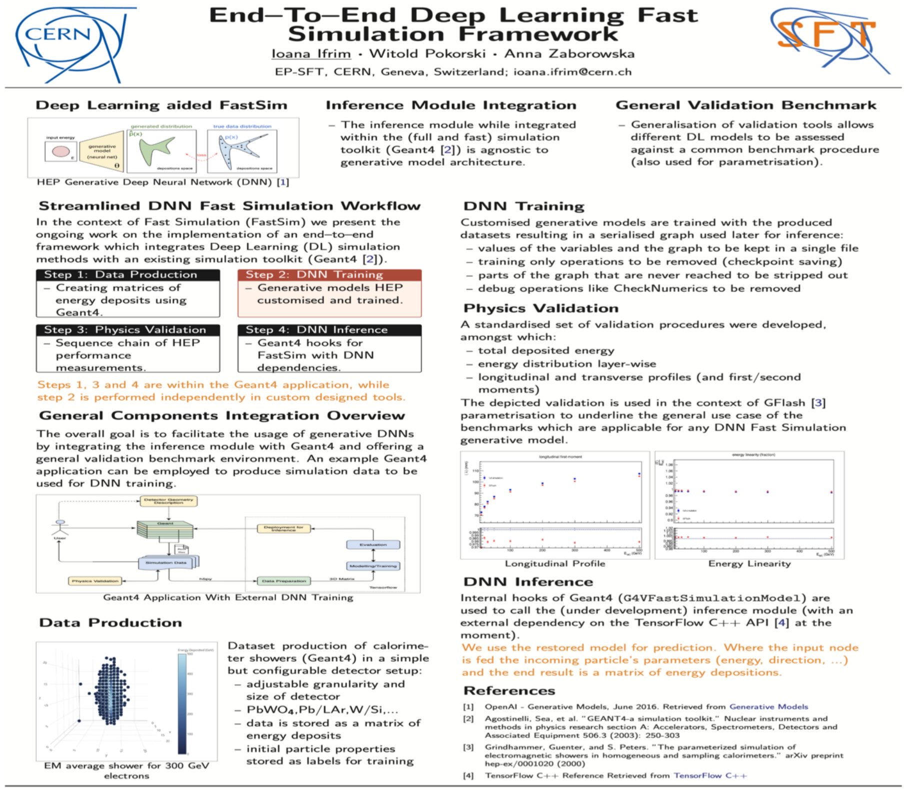




ML-based FastSim in 2019



- Investigated generative models capabilities and limitations for calorimeter shower simulation
    - GAN, VAE, Auto-regressive and customized Auto-regressive network based on a ResNet architecture
- Optimised training procedures through parallelization on GPUs
- Investigated different Loss Functions from deterministic to probabilistic
- Worked the 'full cycle' example/framework implementation
    - Geant4 full simulation -> data processing -> network training -> inference integration (C++ wrapper using Tensorflow C API) -> Geant4 fast simulation



End–To–End Deep Learning Fast Simulation Framework

Ioana Ifrim · Witold Pokorski · Anna Zaborowska

EP-SFT, CERN, Geneva, Switzerland; ioana.ifrim@cern.ch

Deep Learning aided FastSim

HEP Generative Deep Neural Network (DNN) [1]

Inference Module Integration

– The inference module while integrated within the (full and fast) simulation toolkit (Geant4 [2]) is agnostic to generative model architecture.

General Validation Benchmark

– Generalisation of validation tools allows different DL models to be assessed against a common benchmark procedure (also used for parametrisation).

Streamlined DNN Fast Simulation Workflow

In the context of Fast Simulation (FastSim) we present the ongoing work on the implementation of an end–to–end framework which integrates Deep Learning (DL) simulation methods with an existing simulation toolkit (Geant4 [2]).

Step 1: Data Production – Creating matrices of energy deposits using Geant4.

Step 2: DNN Training – Generative models HEP customised and trained.

Step 3: Physics Validation – Sequence chain of HEP performance measurements.

Step 4: DNN Inference – Geant4 hooks for FastSim with DNN dependencies.

Steps 1, 3 and 4 are within the Geant4 application, while step 2 is performed independently in custom designed tools.

General Components Integration Overview

The overall goal is to facilitate the usage of generative DNNs by integrating the inference module with Geant4 and offering a general validation benchmark environment. An example Geant4 application can be employed to produce simulation data to be used for DNN training.

Geant4 Application With External DNN Training

Data Production

EM average shower for 300 GeV electrons

Dataset production of calorimeter showers (Geant4) in a simple but configurable detector setup:
– adjustable granularity and size of detector
– $PbWO_4$,Pb/LAr,W/Si,...
– data is stored as a matrix of energy deposits
– initial particle properties stored as labels for training

DNN Training

Customised generative models are trained with the produced datasets resulting in a serialised graph used later for inference:
– values of the variables and the graph to be kept in a single file
– training only operations to be removed (checkpoint saving)
– parts of the graph that are never reached to be stripped out
– debug operations like CheckNumerics to be removed

Physics Validation

A standardised set of validation procedures were developed, amongst which:
– total deposited energy
– energy distribution layer-wise
– longitudinal and transverse profiles (and first/second moments)

The depicted validation is used in the context of GFlash [3] parametrisation to underline the general use case of the benchmarks which are applicable for any DNN Fast Simulation generative model.

Longitudinal Profile   Energy Linearity

DNN Inference

Internal hooks of Geant4 (G4VFastSimulationModel) are used to call the (under development) inference module (with an external dependency on the TensorFlow C++ API [4] at the moment).

We use the restored model for prediction. Where the input node is fed the incoming particle's parameters (energy, direction, ...) and the end result is a matrix of energy depositions.

References

[1] OpenAI - Generative Models, June 2016. Retrieved from Generative Models
[2] Agostinelli, Sea, et al. "GEANT4-a simulation toolkit." Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment 506.3 (2003): 250-303
[3] Grindhammer, Guenter, and S. Peters. "The parameterized simulation of electromagnetic showers in homogeneous and sampling calorimeters." arXiv preprint hep-ex/0001020 (2000)
[4] TensorFlow C++ Reference Retrieved from TensorFlow C++

Where do we go now?
Plan for 2020

- Fast Simulation
- Geant4 modernization and improvements
- use of compute accelerators R&D

(Geant4 modernization and improvements; use of compute accelerators R&D:) strongly correlated

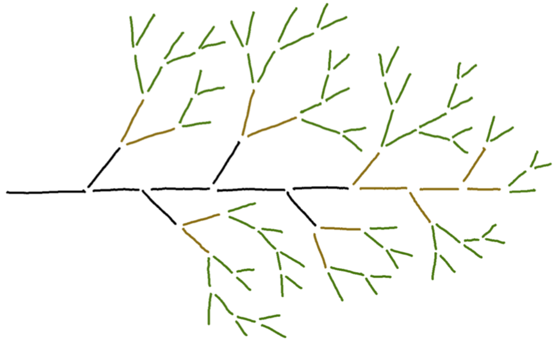
Fast simulation in 2020

‘Classical’ EM shower parametrisation

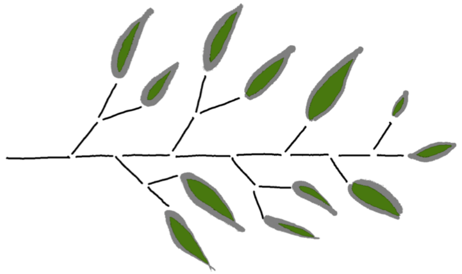
Plan of work for 2020:

- ➔ Finalise transverse profile parametrisation
- ➔ Propose implementation of tools within G4 to automate tuning
- ➔ Develop appropriate examples in different materials, granularities
- ➔ Additions:
  - ◆ Improve transverse profiles
  - ◆ Introduce start of shower parametrisation (as in CMS shower parametrisation)
  - ◆ Investigate parametrisation of only particles below energy threshold (‘core’ of shower: full sim, ‘branches’ parametrised)

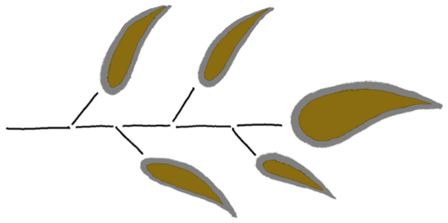
Full shower simulation



Parametrisation of particles with energy $E<E_0$

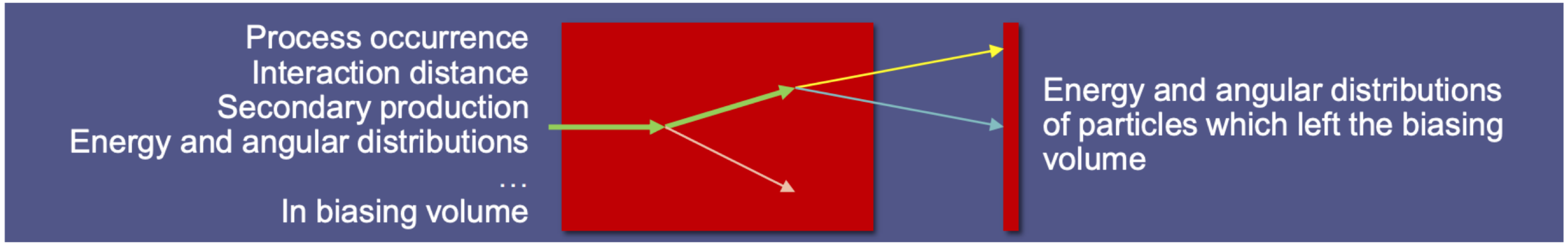


Parametrisation of particles with energy $E<E_1$ ($E_1>E_0$)



Biasing - validation tools

- Many biasing techniques (e.g. Russian roulette, weight window, …) present in G4, with examples, but … need for general validation tools

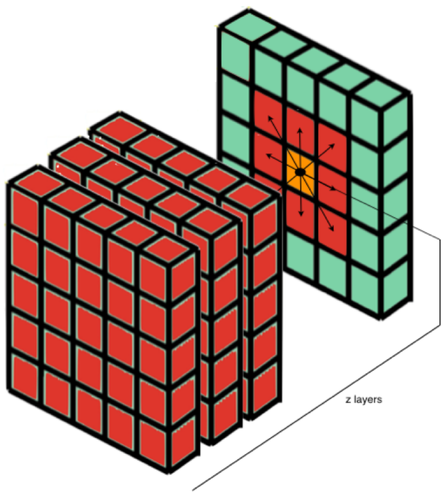


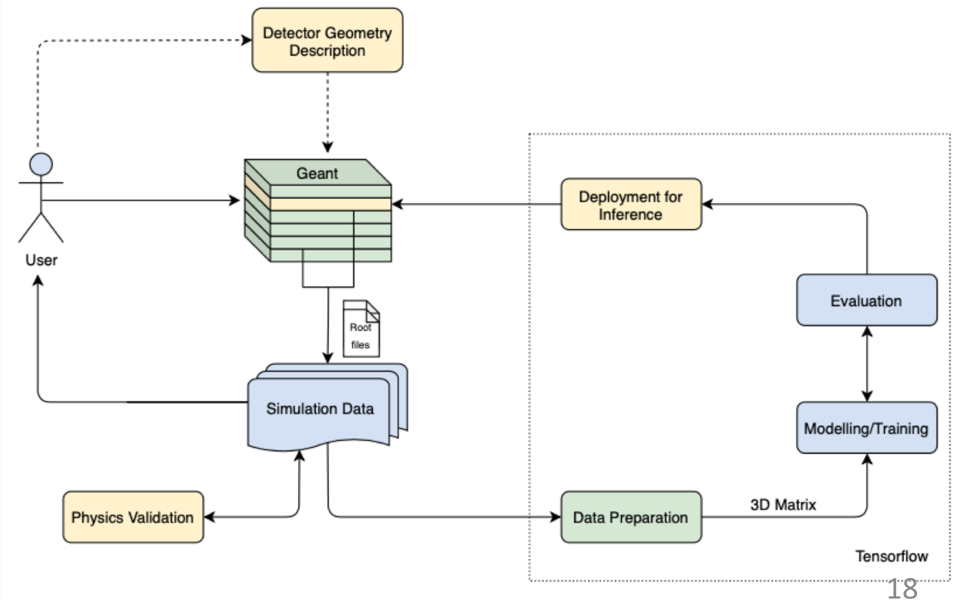
M.Verderi (23th Collaboration Meeting)

- Started as a generalisation of biasing examples (importance sampling, weight window biasing in radiation protection)
- Needs to be improved, extended, documented, and integrated into geant-val

ML-based FastSim in 2020

- Finalize the development of the auto-regressive model
  - Improve dependency model between calorimeter cells (training)
  - Improve data encoding/decoding tools
- Extend and finalize framework implementation for integration with Geant4
  - Automate data preparation
  - Include other generative models types for inference module





Geant4 modernization and use of compute accelerators in 2020

Geant4 modernization and improvements R&Ds

- Geant4 ‘stateless’ transport prototype
  - move the ‘state’ from managers to tracks to allow further study of track-level parallelism and GPU usage
- Automate performance measurements
  - hotspots and microarchitecture exploration data
- Single precision usage in simulation components
  - study numerical stability and changes required
- Instruction and data cache optimizations
  - reordering of data members, group Booleans into bit fields, etc
- Development of compact, self-contained physics library with predefined scoring options (for example for EM shower simulation)
  - would allow, for example, to stay ‘confined’ on GPU for an important % of simulation time

Simulation on accelerators – current status

- Hardware quite different from CPUs
  - Favoring massive parallelism exposed by the software
- Geant4 is a state machine toolkit, evolving states (tracks) per step
  - Unpredictable processing sequence (particle physics is stochastic) -> hard to make scalar processing pipelines
  - A direct porting of Geant4 ”as is” to GPU practically impossible and/or very inefficient
- Existing efforts for porting Geant4 to GPU
  - No available toolkit on accelerators for general purpose simulation
    - Ports only for limited corner cases: optical photons, neutrons, medical physics
  - Ongoing R&D exploration efforts for general purpose GPU porting in US
    - Geant Exascale Pilot Project

GPU R&D #1: start from what we have

- Evolving a GPU simulation model based on a simple prototype
  - Based on existing portable components developed in GeantV context
    - Mostly geometry (VecGeom), but easy to extend to others (magnetic field, physics)
- Simple ray-tracer using geometry only
  - Copy full (VecGeom) geometry to GPU (streaming mechanism working)
  - Writing CUDA kernel making the “X-ray” image of geometry
    - One ray per image pixel -> massive parallelism
- Expand the prototype, adding more models and eventually a simulation flow
  - Field, physics models creating secondaries
  - Trying to adapt the simulation to the device requirements rather than map existing CPU model to device

GPU R&D #2: Adapt simulation to vendor optimized GPU ray-tracing package

- Optix provides a HW-accelerated ray-tracing framework
  - Allowing user-defined geometry
  - Offering optimized scheduling of 'rays' to user-defined kernels
    - 'Shaders' for Optix -> kernels embedding (physics) models for us
- Start with simple prototype based on examples
  - E.g. JUNO optical photon Optix-based simulation (Opticks, S.Blyth)
  - Understand limitations and extension opportunities
- If prototype exercise promising, extend Optix-based model
  - Interact with Optix dev team for possible functionality extensions
  - Extend the prototype to a more comprehensive simulation

GPU R&D #3: Understand a sustainable portability model for device

- Needed at medium/long-term scale by #1, #2 or any other approach
  - Code base is large
- Understand features/limitations of existing performance portability frameworks
  - evaluate frameworks like Kokkos, Alpaka or SYCL/oneApi

- The optimal solution may be in the middle, transforming our code to be more GPU-friendly, then using performance portability tools

Complementary GPU R&D

- Geometry transformations exposing massive parallelism
  - For example: single solid type (polyhedron, tetrahedra), or tessellations (everything made of triangles)

- Track data management to favor pipeline workflows
  - i.e. benefit from both code and data locality
    - E.g flushing large blocks with track data through pipelines of kernels and accounting via masks
  - Easier to explore using the GeantV prototype before embarking into deep Geant4 transformations

Miscellaneous

- Geant4 R&D Task Force
  - coordination and contribution

- EP R&D
  - Fast simulation project on ML-based parameterizations

- AIDA2020 and AIDA++
  - VecGeom, Fast Simulation

- HSF Detector Simulation working group
  - coordination and contribution

Summary

- R&D on simulation software essential to meet the HL-LHC (and post) requirements
- GeantV project concluded with a number of valuable findings
- three axis of further development
  - improve and modernize Geant4
  - Fast simulation R&D
  - compute accelerators usage in simulation
- 2020 will be devoted to prototyping work to identify the most promising directions