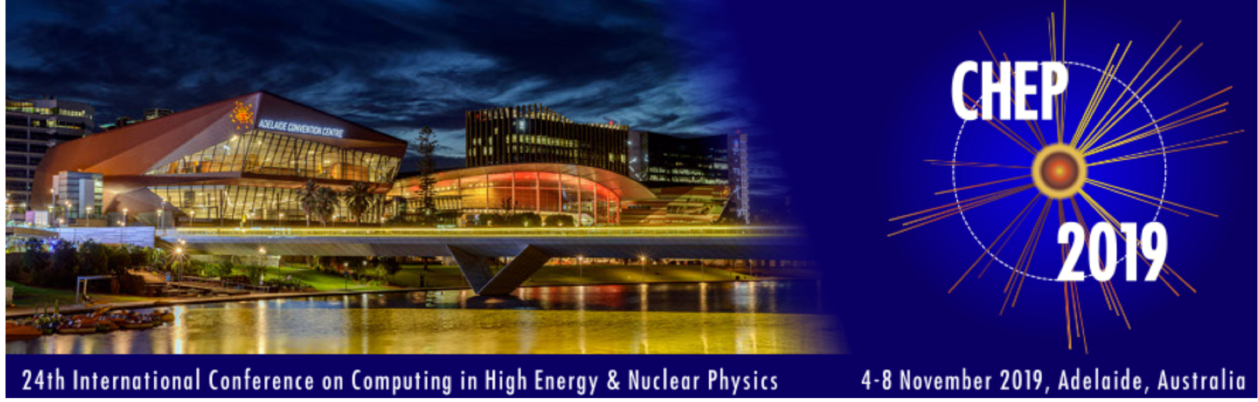

CHEP
2019
24th International Conference on Computing in High Energy & Nuclear Physics
4-8 November 2019, Adelaide, Australia

Evolving Geant4 to cope with the new HEP computing challenges

A Gheata for the Geant4 Collaboration

4-8 Nov 2019, Adelaide

Context

- Increasing requirements for simulated samples towards HL
  - Full simulation still a bottleneck in many workflows
  - Speedup of several factors needed
- Important lessons pointing to a large potential of speedup in Geant4
  - Restructuring and optimizations: more compact and streamlined physics code
  - Vectorizing few FP-intensive modules: field integration, MSC
- Accelerators and HTC become even more important
  - Making simulation code accelerator friendly becomes a necessity
  - Integration in experiment-driven parallel frameworks essential

Geant4 Task Force for R&Ds

- Promote & survey R&D on software architectural revisions
  - Including adaptations to emerging technologies and architectures beneficial to Geant4
- Provide communication/support among/for R&D activities
- Assess performance of different improvements and effort for integration
  - Make recommendations to the Geant4 SB
- Intended as catalyzer for fast performance developments integration
  - Coming from within and outside the collaboration
  - Regular meetings focused on ideas & follow-up of R&D activities

Performance: main directions

GEANT4
A SIMULATION TOOLKIT

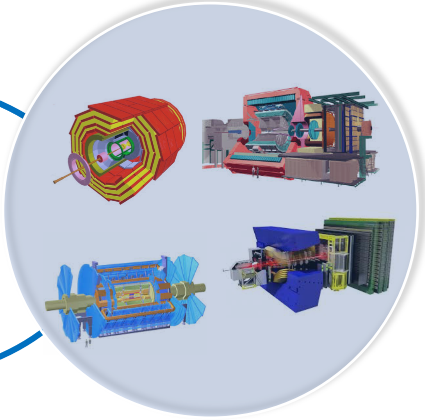
HPC/HTC
CPU - accelerators

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

Integration

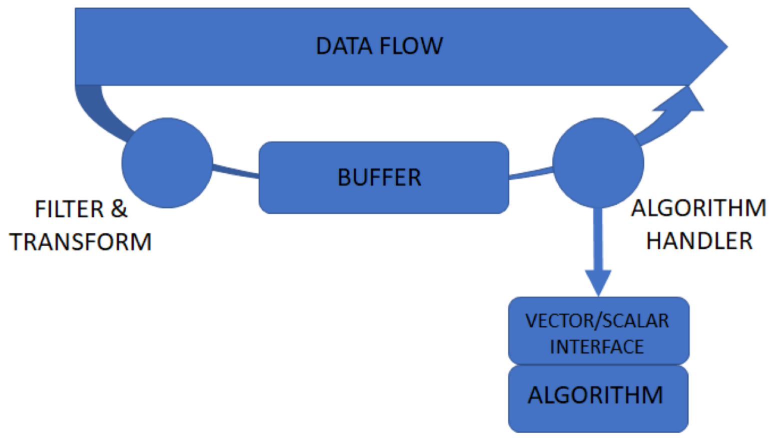


Fast sim
Parameterizations - ML

Investigations of structural changes

- Code compactness and simplified calling sequence
  - A large part of the GeantV speed-up coming from better fitting the instruction cache
  - CPU μ-pipe for complex Geant4 simulation largely front-end bound
- More streamlined computation for HEP simulation hotspots
  - E.g. EM shower or neutron physics
  - Confined “per cell” simulation w/o callbacks
- Data structures and flows adapted for accelerators
  - Efficient track-level parallelism on warps
- Ongoing R&Ds / discussions, targeting concrete prototypes

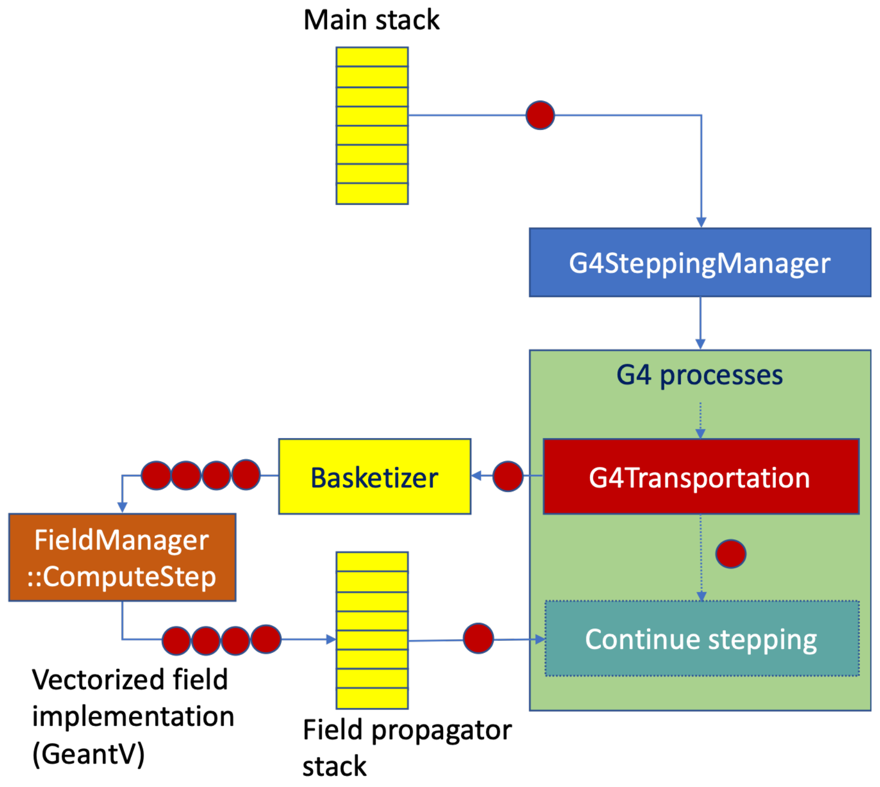
Vector pipelines in Geant4



VectorFlow: A vector adapter

A way to generalize vectorization by passing vectors of data to functions rather than rely on inner loops.

- Idea originating from GeantV workflow, but generalized as templated API usable in any workflow
  - Using VecCore as vectorization library
- Prototyping the changes needed in Geant4 for such extension
  - Ongoing work for making Geant4 transport stateless
  - Aiming to prototype integration w/ FP-intensive modules



A. Gheata (CERN), W. Pokorsky (CERN)

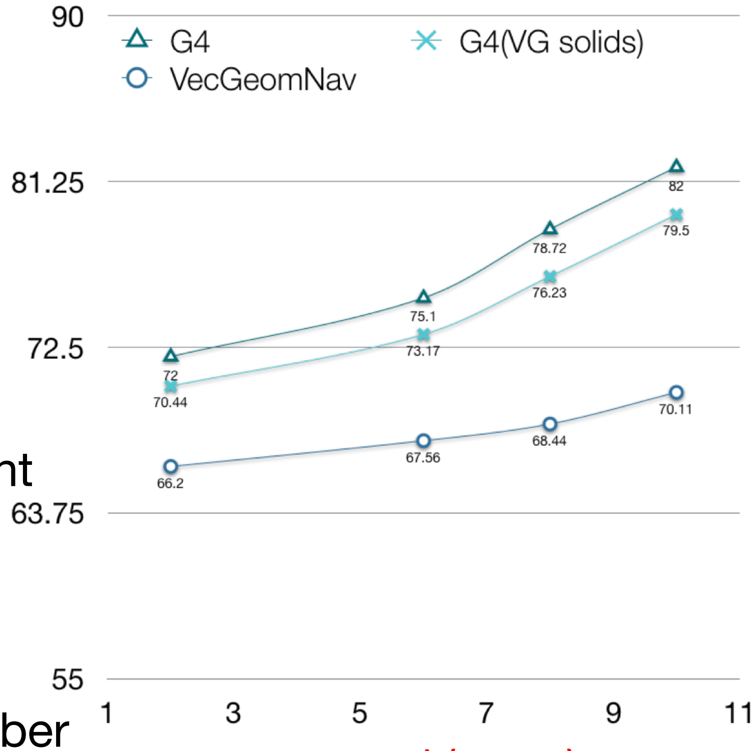
Optimizing Geant4 navigation using VecGeom

VecGeom

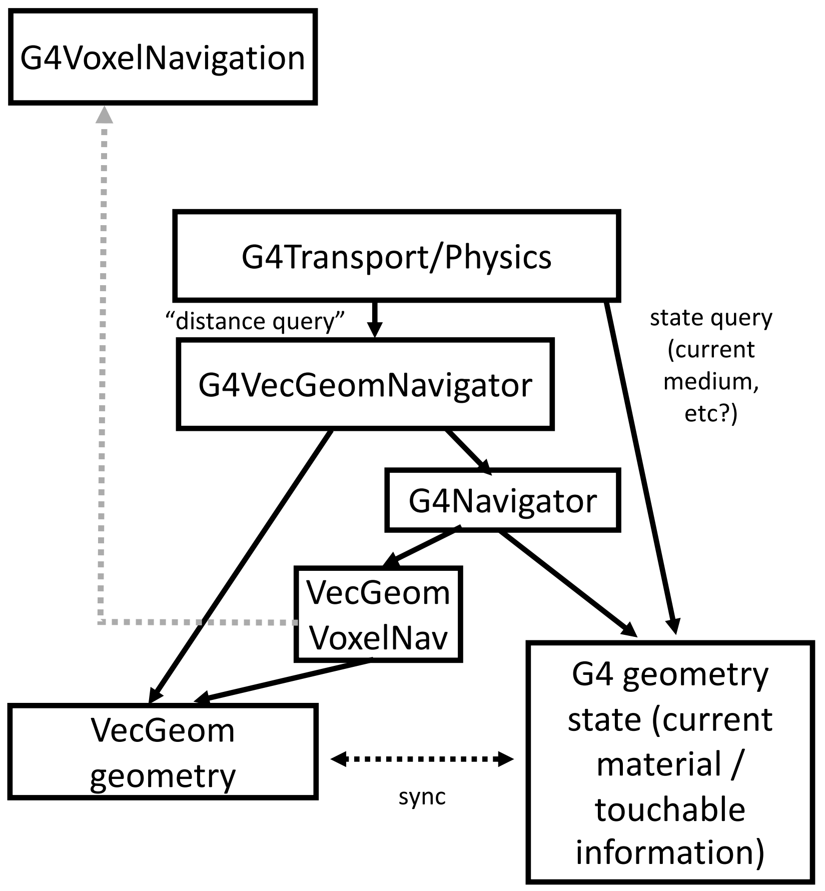
See talk of Sandro

- a first implementation of a Geant4 **navigation plugin**, using VecGeom capabilities.
  - allows to make use of the modular and extensible navigation acceleration structures of VecGeom
- Tests on full detector geometries remain to be done and development to be completed
- Related ongoing work: VecGeom navigation specialization for a number of volume topologies

First tests and evaluations



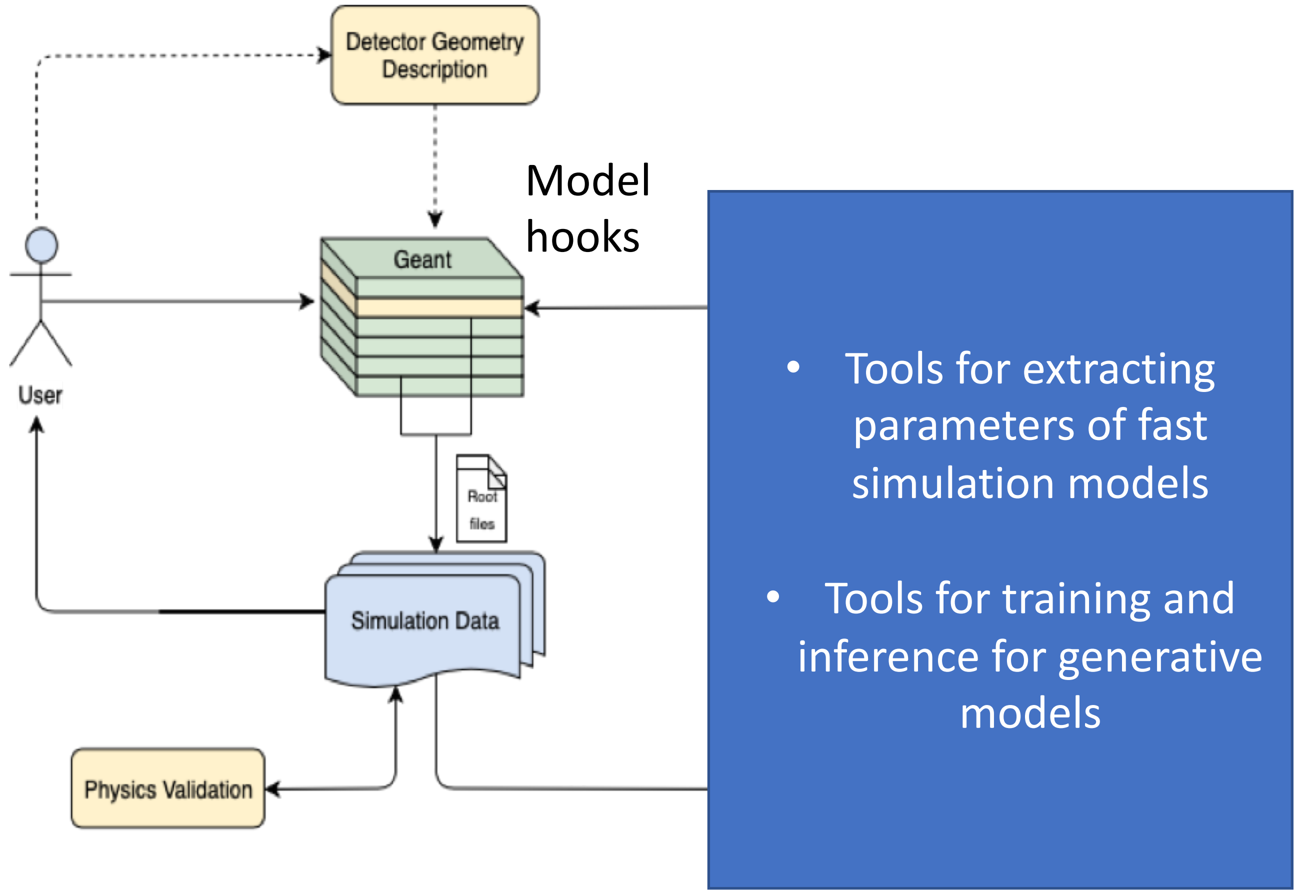
S. Wenzel (CERN)



One of the implementation approaches

Automated tools for fast simulation

- Fast simulation is experiment dependent
  - custom procedures to extract parameterisations
- Ongoing work for streamlining the procedure
  - ➢ Users come with their own setup
  - ➢ Full simulation producing standard information (hits)
  - ➢ Simplified, automatic and easy to use procedures to extract the parameters (e.g. fitting shower profiles, training networks)
  - ➢ Plugged back in simulation via Geant4 fast sim hooks
- On-demand add-ons to Geant4 installation



- Tools for extracting parameters of fast simulation models
- Tools for training and inference for generative models

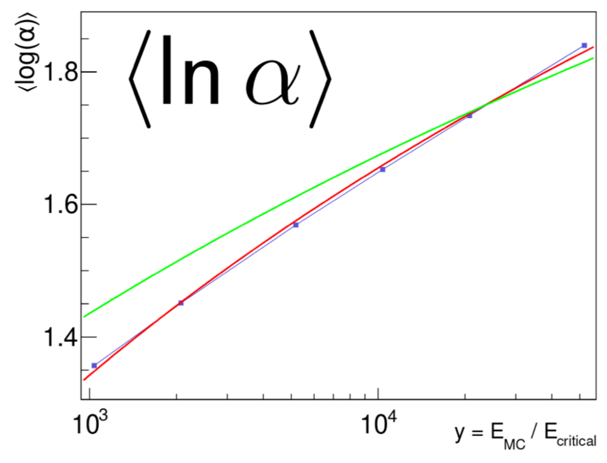
Automating parameterisation for fast simulation

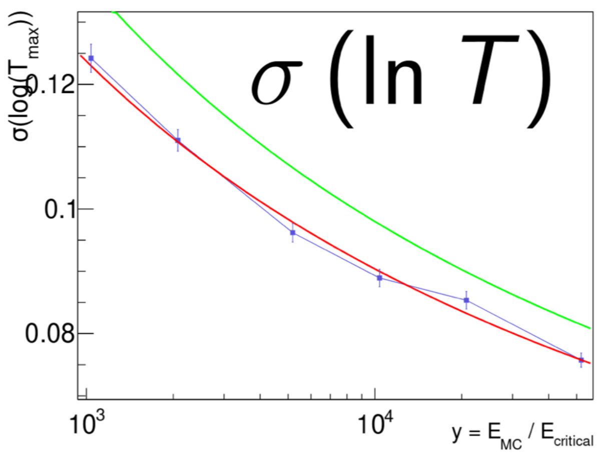
Revisiting fast simulation hooks and models offered by Geant4 (e.g. GFlash).

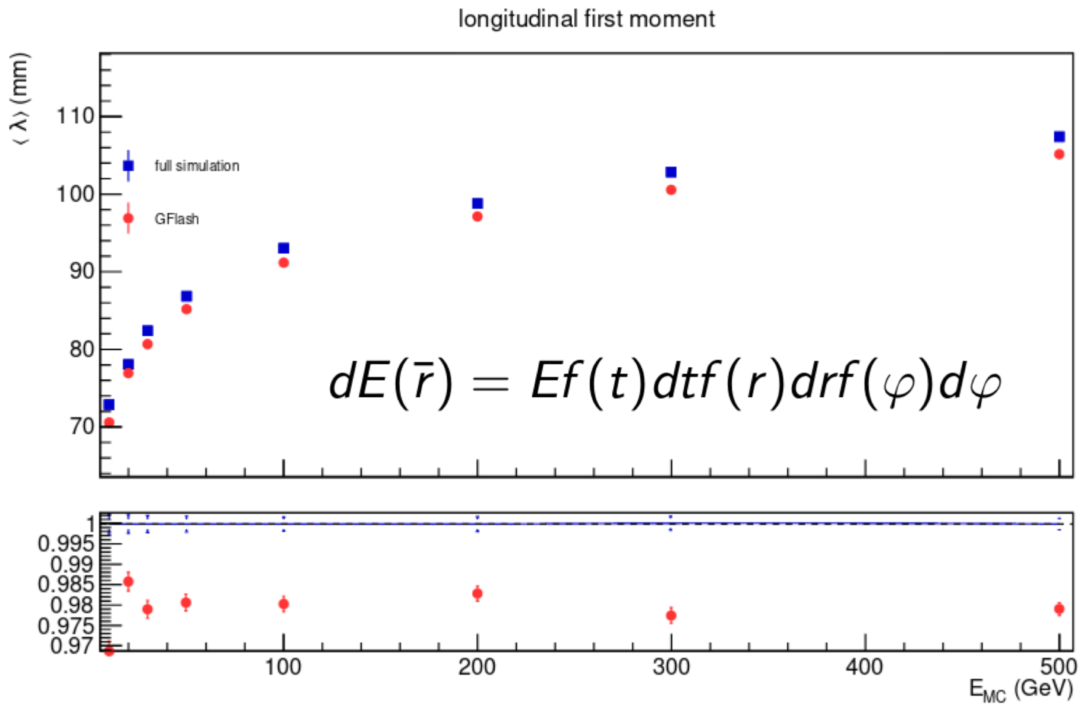
Work on automation of EM shower parametrisation - tuning of parameters for users' geometry.

More validation tools developed (for fast simulation, biasing - comparison to standard simulation)

A. Zaborowska (CERN)





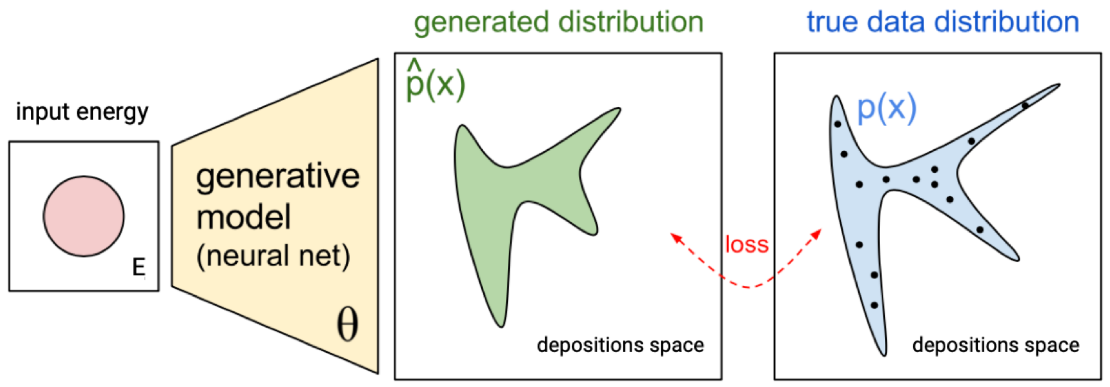


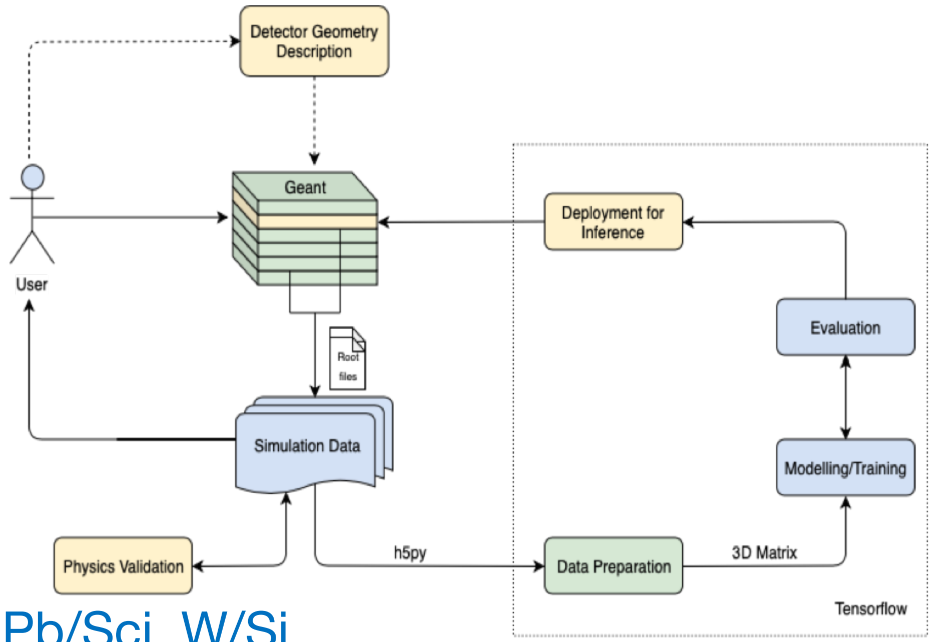
$$dE(\bar{r}) = Ef(t)dtf(r)drf(\varphi)d\varphi$$

green line - GFlash parameters from arXiv:hep-ex/0001020
blue points - from full simulation on Pb
red line - fit to full simulation (e.g. no Z dependency)

Machine Learning Approaches

See talk and poster of Ioana





I. Ifrim (CERN)

- Training on simulated data
  - Different calorimeter geometries : PbWO4, Pb/LAr, Pb/Sci, W/Si
- Different architectures are tested
  - Autoregressive Networks, Variational Auto-encoders, Generative Adversarial Networks
  - Validation against MC truth
- Training/inference workflow aiming to integrate with Geant4 hooks
  - Collaboration between SFT & openlab (CERN) w/ close contact with experiments

https://openai.com/blog/generative-models/

Task parallelism in Geant4

J. Madsen (LBL)

- Geant4 can benefit from having internally nested task-based parallelism
  - Making parallelism transparent to users (i.e no G4MTRunManager)
  - Better support for sub-event parallelism, eventually track-level parallelism
- Easier to expose simulation as a task
  - In relation with concurrent task based frameworks (e.g. CMSSW, Gaudi, …)
- First implementation of tasking support already available
  - gitlab.cern.ch/jmadsen/geant4-tasking
  - Based on standalone tasking library: github.com/jrmadsen/PTL
    - Native C++ features (future, promise, packaged_task, coroutines)
    - TBB backend available, PTL forwards task to TBB scheduler instead of internal
  - Support for multiple task pools
    - E.g for off-loading work to coprocessors

Other optimizations and new features

- Magnetic field class refactoring
  - Transforming runtime into static polymorphism w/ more inlining
  - Multi-particle driver integration
- Sub-event parallelism
  - Allow splitting events in vertices processed by separate threads
  - Better scaling and MT workload for large events
- Refactoring transportation
  - Splitting transportation process in “flavors” dealing separately with:
    - Charged/neutral particles, optical photons, parallel geometries or other specialized cases
  - Increasing code locality and decreasing branching

Geant Exascale Pilot Project

See talk of Liz

- Goals
  - Investigate how to best use GPUs for full HEP simulation
  - Explore memory access, computation ordering, and CPU/GPU communication patterns
  - Avoid over-simplification
- Collaboration
  - Fermilab, Lawrence Berkeley Lab, Oak Ridge National Lab, participants from US-CMS, US-ATLAS
- Strategies
  - Reuse or leverage existing packages, not bound by backward compatibility.
  - Focus on **NVidia** compiler at first (later look at Kokkos and others)
  - **Partial Static Polymorphism**: allow upload/download of data to device without transformation
  - **Separation Of State and Access and Functional Approach**: allow significant data memory layout change without code change
  - Research way to increase instruction and data cache efficiency

P. Canal (FNAL), J. Madsen (LBL) et al

Conclusions

- Geant4 puts very high priority on performance related developments
  - Task force for R&D as catalyzer for development & integration
- Several performance improvement directions being followed
  - Structural changes & optimizations, but also fast sim workflows integrated with Geant4
- R&D on extending the parallelism model
  - More efficient integration with experiments parallel frameworks
  - Targeting also accelerators and HTC
- Aiming for accelerated prototyping/integration cycle