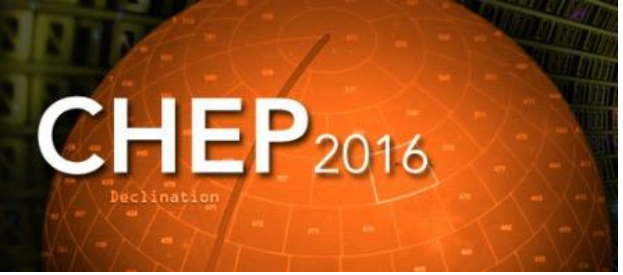
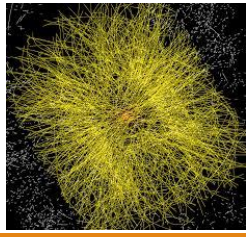


Simulation, Key Issues for the Coming Decade

FEDERICO CARMINATI

CERN-SFT



Disclaimer

I took my mandate for this talk to identify which aspect of simulation leaves room for improvement and how I think it could be improved

This (also) means to put the accent more on the negative than on the positive side of things

Although the issues I raise are quite general for the simulation field, most of my experience is with the GEANT family of codes (2,3,4,V) and this may have introduced a bias

Acknowledgements

For this talk I have received an impressive amount of feedback and suggestions from colleagues

Of course I take full responsibility for the content and all errors or misunderstandings are mine…

The panorama

EGS 1,2,3,4 (1965, ~1970, 1978, ~1990)

- e- gamma. It has set the basic structure of the transport MC
- EGSnrc for medical applications released in 2000 (C++ geometrical library)

FLUKA (1962)

- Complete particle transport solution

GEANT 1,2, 3 (1974, 1976, 1980)

- HEP standard for detector simulation before 2000

GEANT4 (1998)

- Current HEP standard for detector simulation

GEANTV

- Prototype of the *future generation* GEANT

The panorama

MARS (1974)

- Biased transport and beam simulation

MCNP(x) (1957)

- One of the “bibles” of simulation. Now a closed source code

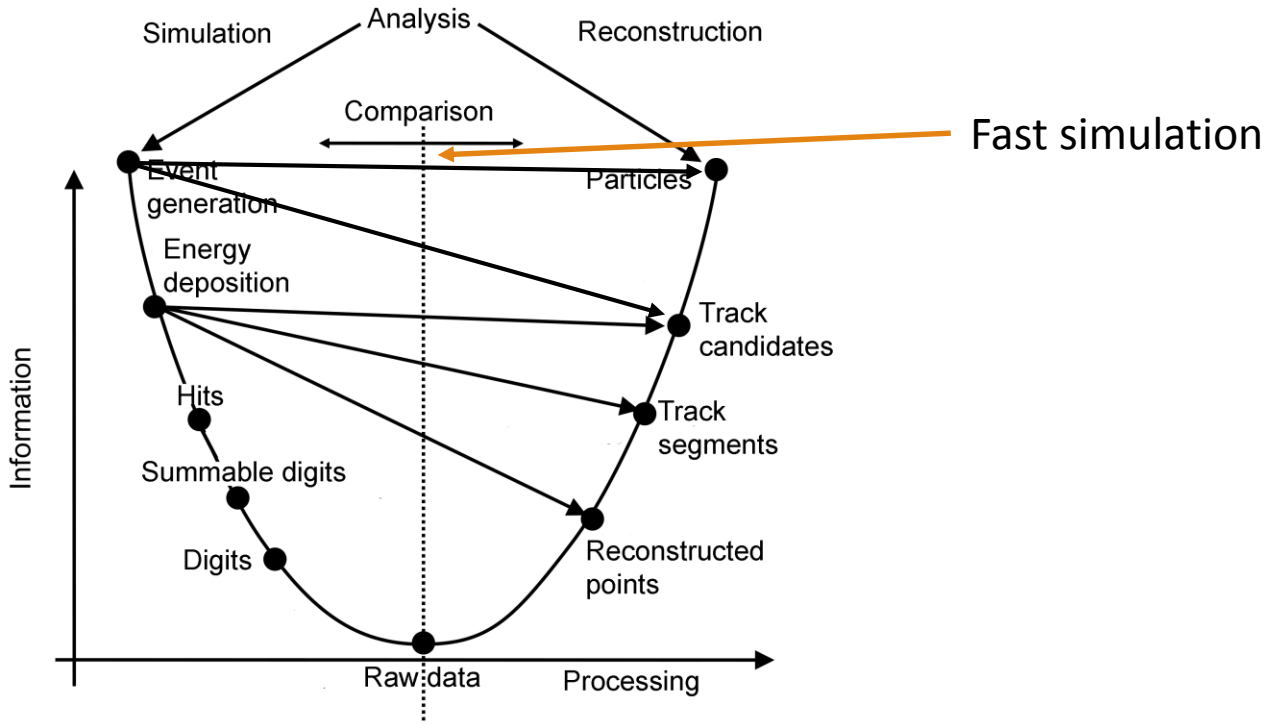
PENELOPE (1996)

- High-accuracy for coupled electron-photon transport

PHITS (2002-present)

- NMTC/JAERI97 all-particle transport with extra capabilities for heavy ion transport

The simulation cycle



Present situation

Two main simulation products used in HEP today: FLUKA and GEANT4

- FLUKA is the main tool per accelerator, beam-machine interactions, targets and secondary beam design, radioprotection & waste prediction and management
- GEANT is the program of choice for full-detector simulation

Each one has its strengths and weakness

- Which I will not discuss here

Collaboration between these two entities is limited

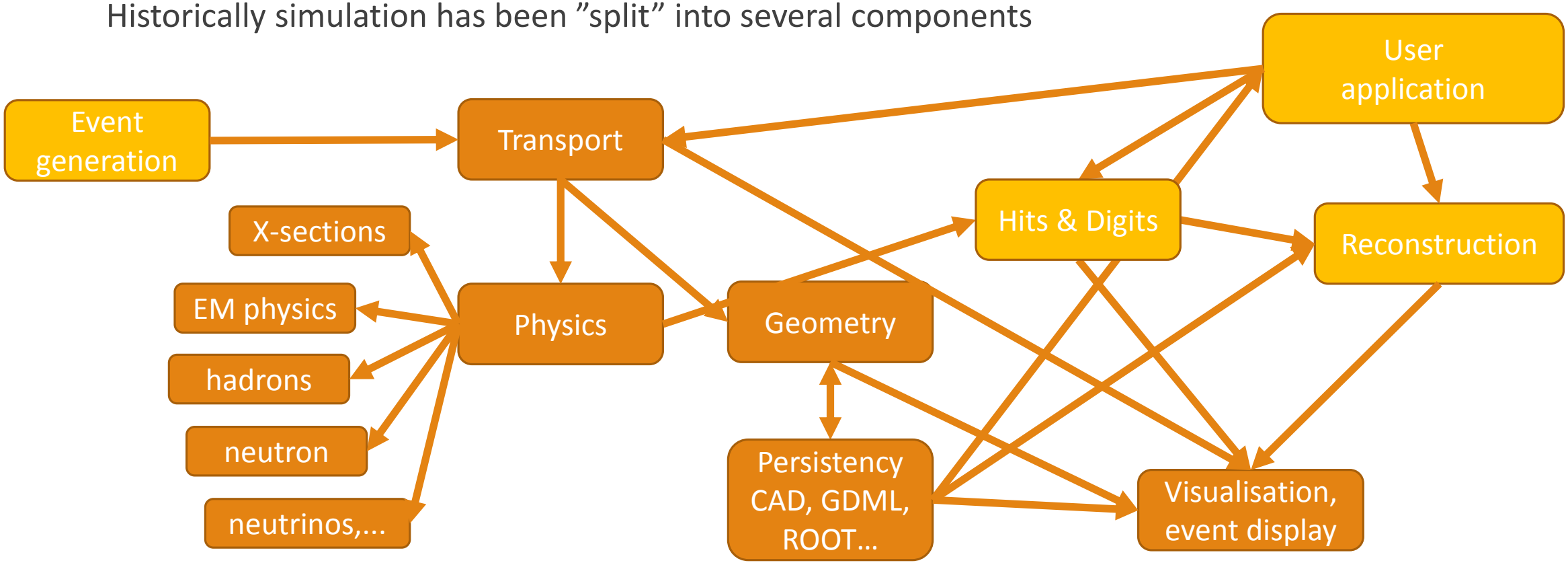
I believe there are missed opportunities here

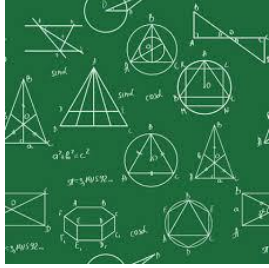
- But I do not see the situation changing anytime soon

The upside is that we have independent confirmation of results and scientific competition, which are positive elements

The components

Historically simulation has been ”split” into several components





Geometry

Geometry is ubiquitously used in HEP (simulation, calibration, reconstruction, alignment, visualization, fast simulation)

- Navigation algorithms can be used in other contexts
- A large development and maintenance effort

Still no simulation package has “by design” provided a ”modular” geometry engine that can be easily reused in other contexts

“Generic” geometry projects give priority to an “exchange format” and several packages that can read from it

Pros:

- Each package can specialize in its own domain of application

Cons

- The wheel has been reinvented many many times

One becomes two, two becomes three, and out of the third comes the one as the fourth.
Maria Prophetissa (3rd century AD)

Convergence (at last?)

USolids was developed to unify TGeo and GEANT4 geometry packages (or at least shape algorithms)

VecGeom (AIDA I&II projects) code has been developed for GEANTV vectorised transport

Now VecGeom algorithms are available both to GEANT4 and to TGeo

- The USolid interface has been maintained, so the user should see no difference

VecGeom has the potential to introduce a %-level gain for GEANT4

- Algorithm improvement and (internal) vectorisation of some shapes
- It might well offer substantially more benefit if its navigation system is used
- See S.Wenzel's talk "Accelerating Navigation in the VecGeom Geometry Modeller" 12 Oct, 12:00 Sierra A

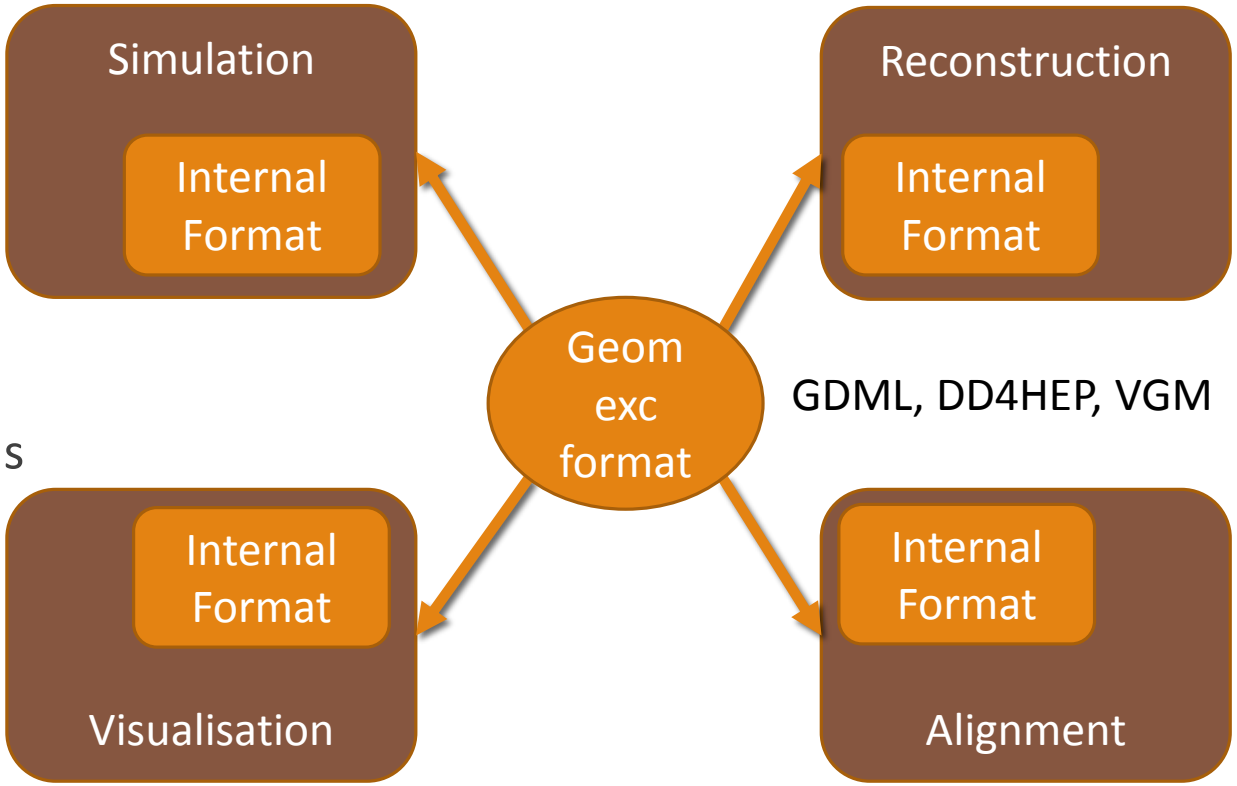
Common exchange format

This solution has been prototyped several times

It is a step in the right direction but

It produces a proliferation of readers and a duplication of algorithms

It hinders community efforts toward solid geometrical algorithms since the code remains in the experiment framework



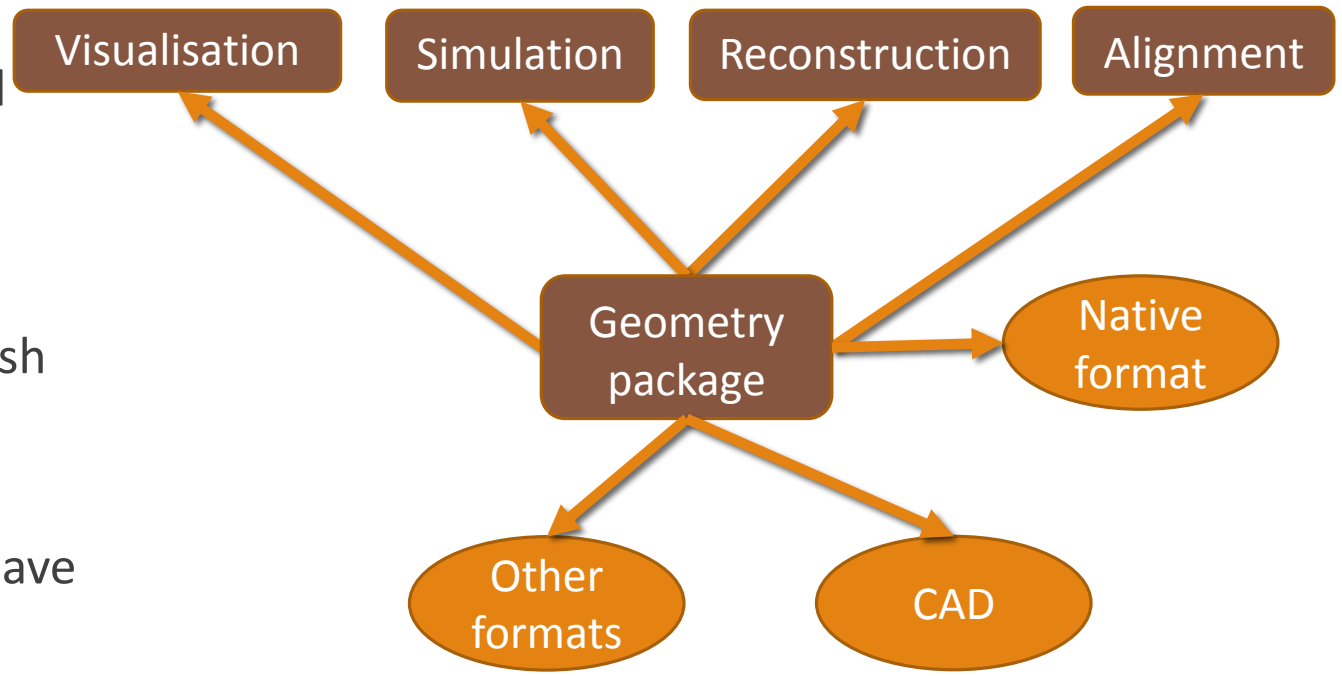
Common geometrical library

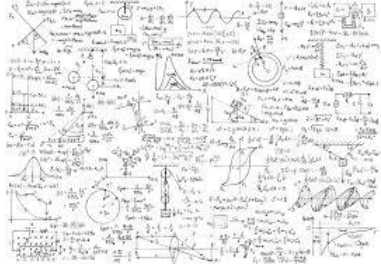
Developing a geometrical modeler is one year of coding and ten years of debugging…

A HEP community geometry package could help concentrate efforts on common algorithms and format convertors

Concentrating on a single, standalone and modular package could avoid the many fresh starts we have witnessed

It could also allow to address the interface with CAD systems, for which all packages have partial solutions





Physics

As precision experiments look for rarer processes, the requirements for their accurate description increase

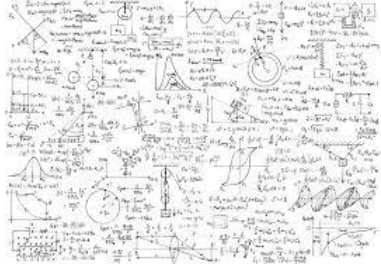
Scarcity of (stable) (wo)manpower sometimes makes planning dependent on the effort available (in time and quality)

- More physics experts with would allow better organization and planning

(In GEANT4) often developments are “user driven” and in “reaction mode”

- It has worked well, but not without some “close calls”

Transport MC physics experts are very few and they not always have the “institutional mandate” to develop physics for MC



Physics

It is now the case, namely in GEANT4, that a generation of extremely skilled developers are approaching the end of their career

Although the problem is known, there is no community-agreed community to address it

Institutions invest heavily in GEANT4

But it would help if there was clear institutional commitment to maintain / develop a given area of physics

Electromagnetic

Supposedly the “simple” part, since QED is well under control

Reinvented several times (EGS, FLUKA, GEANT3, GEANT4, GEANTV)

- Sometimes with more or less “inspiration” (and code) taken from previous versions

The devil is in the details

- Tight connection with transport due to continuous / discrete (energy loss / ionization) processes and multiple scattering
- Low energy (<1keV) and very high energies (>10TeV) require completely different treatment (nuclear shell and molecular effects, high-energy suppressions, nuclear em interactions…)
- High accuracy threshold modeling (higher-order QCD corrections) is mandatory for ILC, CLIC and FCCee (e.g. ttbar), both for generators and for the transport code
- In FLUKA for instance the Compton model evolved from 5 lines to sample Klein-Nishina to 3000 lines to take into account shell-by-shell electron motion!

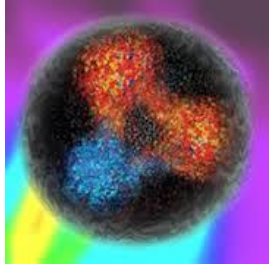
Transport in EM field add an additional dimension of complexity that is still an open field for R&D

Electromagnetic

Often attention to this area is devoted only in case of problems

However developments may take a lot of time, since all ”low hanging” fruits have been harvested by now

It would be important to continue R&D to prepare for the future requirements in terms of precision and rare processes, even if the current codes are quite appropriate for the present needs



Hadronic

Four main “energy regions” are needed with different features and models

- Hadron hadron interaction, intra-nuclear cascade, pre-equilibrium stage, evaporation
- Critical to ensure coherence in the “overlap region”

Different models can be available for each region

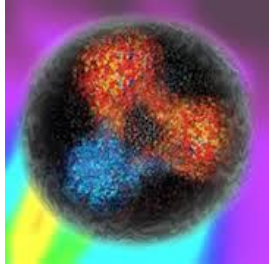
Very few experts worldwide

Mixture of “model based” and “parametric data tuning”

- However if the tuning is done correctly and respecting known conservations, it could provide a sound (and predictive) code

Sometimes we have a “butterfly effect”

- Any change (even an improvement) can have unforeseen effects on the result



Hadronic

Give precedence to thin target or shower benchmarks?

- "Perfect" thin targets with "perfect models" should ensure good shower reproduction

Sometimes either the model or the thin target data are faulty

- A "good shower" does not guarantee correctness of simulation
- A "bad shower" does not give you almost any hint of what is wrong

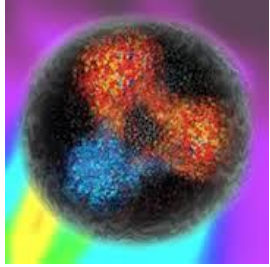
Almost infinite number of possible validation plots

- Inclusive, exclusive, angular, spectra, shower shapes

Even "theory-based" models have several "tunable" parameters

Higher energies mean that we will need to describe the nuclear interactions of more short-lived particles

Trigger-less experiments will also require a precise description of the time structure of the showers



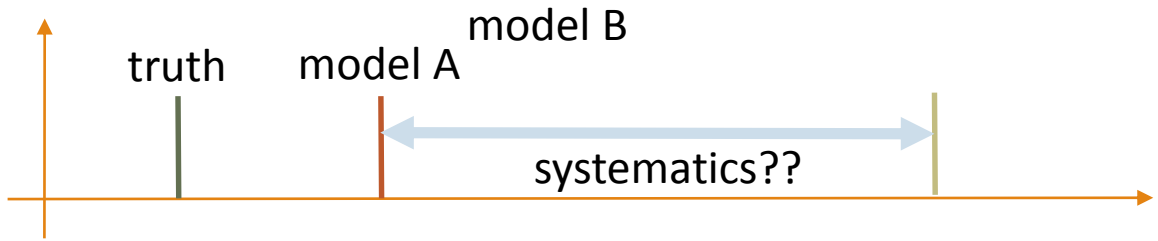
Hadronic

Same remarks as for electromagnetic, the future has to be prepared with a lot of advance

A new, coherent community-driven effort in this area could be beneficial in addressing future challenges

Systematics should be addressed in a more coherent way

- Running two models does not give <u>any</u> information on the accuracy of the results
- Other domains have already moved to full sensitivity analysis (generators)



Fast simulation

Essential to satisfy the foreseen simulation needs of HL-LHC

Rarely introduced “structurally” in the MonteCarlo at the beginning

- GEANT4 had some hooks since the beginning but with few adopters
- Used by CMS, and partly by ATLAS

Considered “too experiment specific” to go beyond some “hooks” in the MC

Usually prompts the development of ”experiment specific” frameworks that either

- Try to “second guess” the MC event loop and are ”maintenance heavy”
- Or are completely disconnected from the full MC

Fast simulation

An *ab initio* design integration would have helped a better synergy between experiments

- See A.Zaborowska's talk *Full and Fast Simulation Framework for the Future Circular Collider Studies,* Track 2, 11 Oct, 14:30

Even if FS is often very experiment dependent, a common framework for FS would probably go a long way in helping experiments

- Provision of common algorithms and tools
- Provision of a common framework to move in and out of fast simulation according to the particle type and the detector

Or at least… (weaker form)

- A good example on how to implement this for a given experiment

Exploring machine learning is also important in this field!

- See O.Shadura's talk *Stochastic optimisation of GeantV code by use of genetic algorithms*, Track 2, 11 Oct, 11:45

Infrastructure

Simulation is often the first application developed in an experiment

Since simulation codes tend to offer a ”framework” (at least architecturally if not functionally, since they handle the event loop) this requires the development of a different ”framework” for reconstruction, reducing reuse and modularity

A more modular approach to simulation would help reuse of the simulation code and would give more freedom to experiments to define their own framework

Speed speed speed

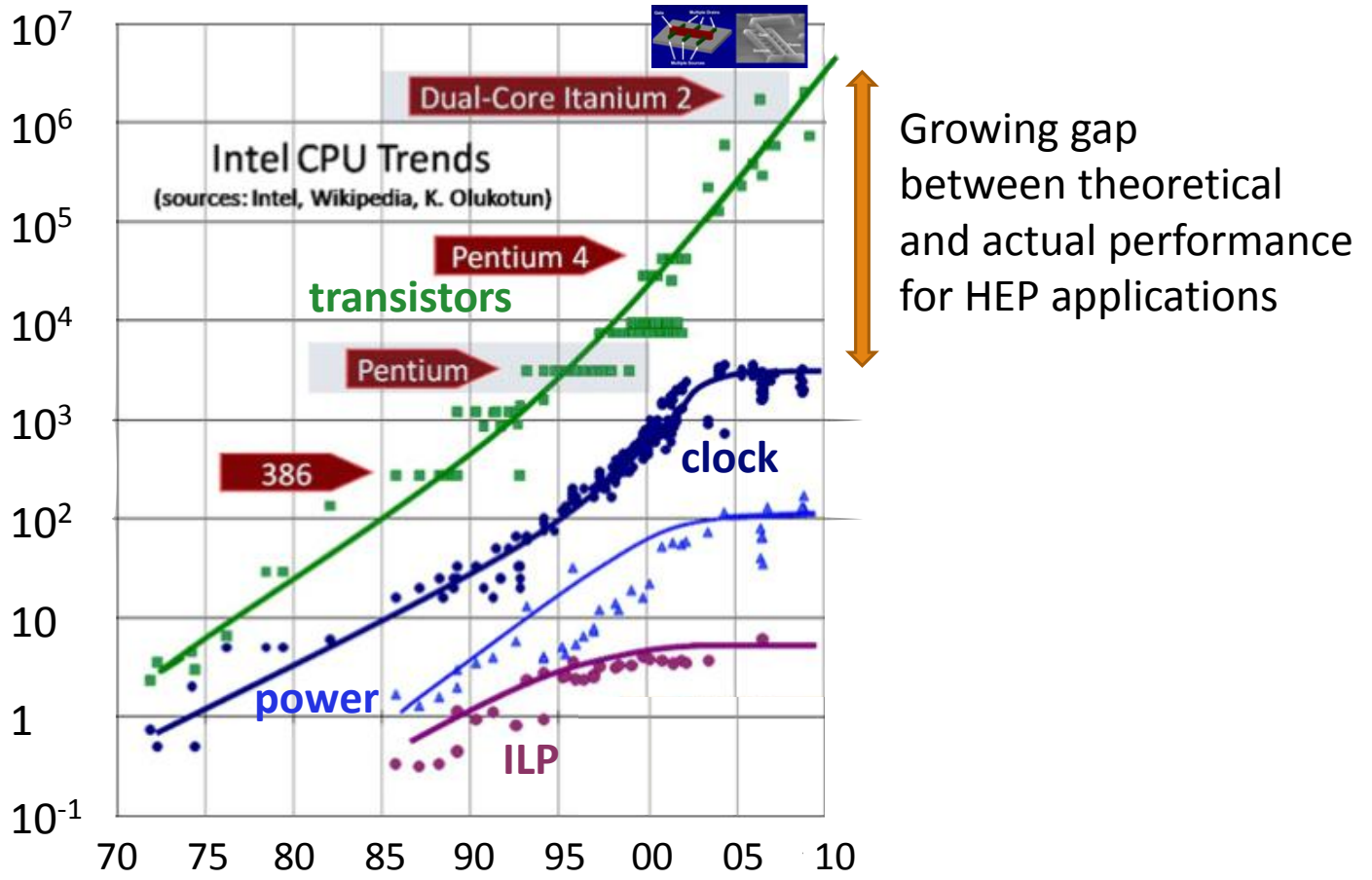
Precision is inversely proportional to the sqrt of the number of events

And needs are projected to grow

The only way to get speed is to exploit SIMD capabilities

And use accelerators...

This (IMNSHO) requires a complete re-write of the code (see *GeantV phase 2: developing the particle transport library*, Track 2, 11 Oct, 16:30)



Growing gap between theoretical and actual performance for HEP applications

Portability

Back in the 80's CERNLIB was maintained on machines with 32, 48, 60, 64, and 66 bit words with different endiannes and byte size, each one with its "idiosyncratic" compiler

For several years now we lived an easy life thanks to Intel and IEEE floating point

The environments of the future will be a lot more heterogeneous

Writing portable code (a lost art) becomes (again) absolutely essential.

This introduces new dimensions to architectural design and testing

Language of course helps a lot, but if you want to reach for the best performance, you will always be a step ahead of what language can offer

Validation – regression

Given the distributed nature of development, validation is also distributed / inhomogeneous

- Developers have their own validation environment, often private
- The situation with “thin target” tests is very diverse (some excellent, some less)
- The “golden standard” for validation are experiment test beams, but they are often impossible to “reproduce”
- Unit tests are scarce / absent (and arguably difficult to implement anyway)

“Automatic alarms” for physics regression could probably help to increase the number of tests we can run

- But there is a steep learning curve to implement them

Validation – regression

Other community have a set of “sanctified benchmarks”

- See for instance Shielding Aspects of Accelerators, Targets and Irradiation Facilities https://www.oecd-nea.org/science/wprs/egsaatif (SATIF)

Some activity (FNAL-CERN-SLAC) started in the direction of providing a consistent Database of validation data in a “MC friendly” way

- See “Software Aspects of the Geant4 Validation Repository”, Track 2, 13 Oct, 16:30

We have to do a better job with experimental data

- Make sure that test-beam data can be used for validation beyond the test-beam lifetime
- Use more high-quality data from experiments in our regression

Biasing

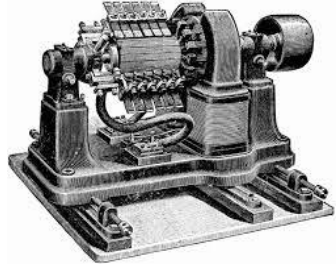
Some codes have a long-established biasing infrastructure (FLUKA, MARS, MCNPx)

Others are relatively new to biasing (GEANT4)

Mixing biasing and “analog” simulation is an ambitious proposition with which we have little experience

Biasing could substantially reduce calculation time in some situations (if used judiciously, very risky otherwise)

It should however be built into the design of the code from the start



Generator interface

The gestation of HepMCx has been long and difficult, but the very fact of having a standard is a very good thing per se

- Even difficult problems have a “common” solution with enough patience and determination

There is still a deep divide between “generators” and “hadronic models” which should and could be bridged

- Sometimes two versions of the same code are used, one for the “generation” of the events and the other for the “nuclear interaction model”

Physics lists

GEANT4 has formally introduced the concept of “physics list” which was “somehow” there in GEANT3 and FLUKA

Not one of the “easiest” items to handle in GEANT4

- It has been perhaps too much “exposed” and “publicized” to users

We now have few “sanctified” physics lists

The flexibility of the physics list is a powerful instrument if used wisely

Very useful concept also to mix fast and full simulation

Having different models easily interchangeable is great

- but… see previous discussion on “systematics”

Outside HEP…

The list of applications outside HEP is very large

- Design of medical instruments
- Radiation treatment planning
- Radiation safety (space, accelerators…)
- DNA damage, microdosimetry

In general these are considered “highly welcome users” and we largely profit from the reciprocal feedback and developments

- Possibly ~50% of GEANT4 users are outside HEP

However more coordination of these activities between them could be useful here

- Particularly in the medical domain where we really make a difference and our impact could even be greater

Outside HEP

In the GEANT4 case some of these activities are highly coordinated internally

- G4NAMU (G4 North American Medical Users -even if now is from all over the world-) has at least a yearly meeting
- The majority of the tutorials outside of US are organized by non-HEP
- The “Geant4 Space User Workshop” is held approximately every 18 months
- Geant4-DNA is even advertised as a “collaboration” on their own

A large fraction of the hadrotherapy treatment planning systems in Europe are based on FLUKA (in particular for the carbon ions)

Still synergy between these activities and HEP could be improved

Us (simulators) and the others

Simulation codes tend to be self-contained

- Even CLHEP, which is an external package, ships with GEANT4 for convenience

Relations with other packages is usually limited

- Reduction of dependencies ensures an easier installation and use
- It also introduces globally duplications and reinvention of wheels

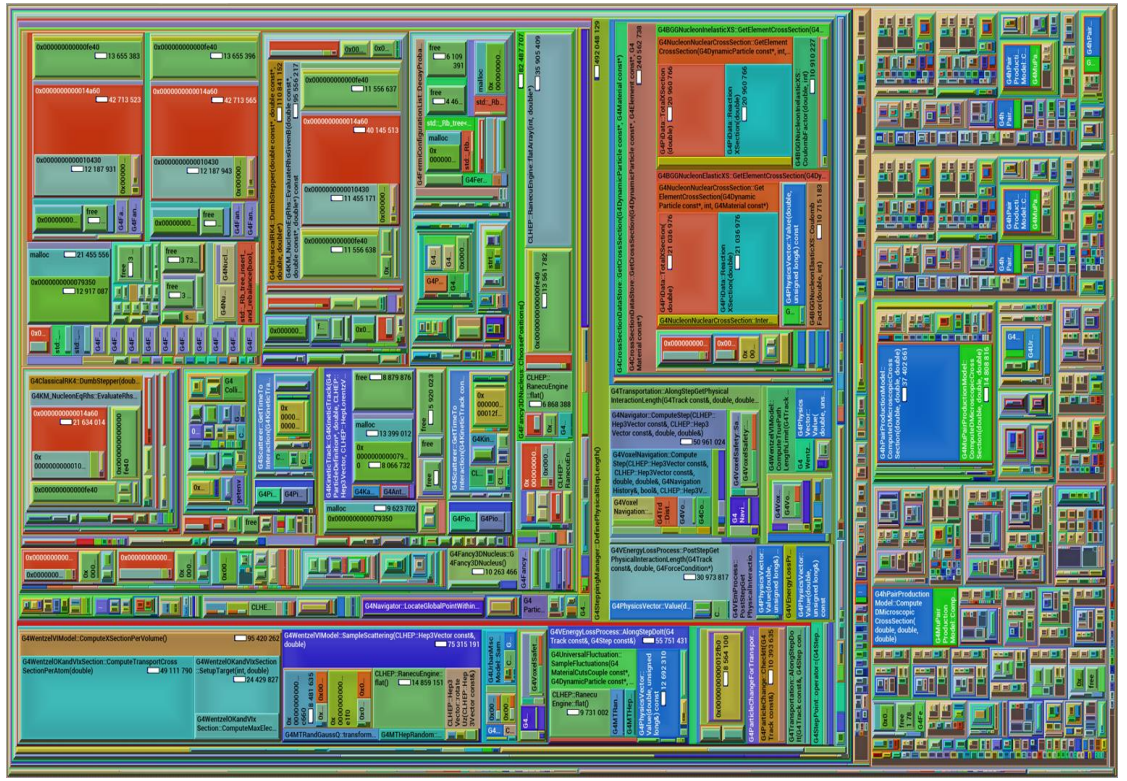
At CERN, ROOT and GEANT4 have been in the same group for more than 10 years but their development has been largely parallel

The difficult balance between the advantages of code reuse and the problems of managing dependencies has been usually decided in favor of monolithic distributions (if not frameworks)

A path toward the future

The need to rewrite the code in order to take advantage from the new architectures has lead to the GeantV project

We think this is a good occasion to address the issues mentioned before and to bring together the community



The carpet

Goals of the GEANTV project

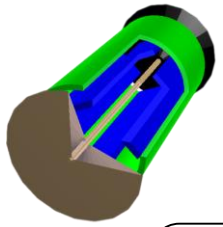
Develop an all-particle transport simulation programme with
- A code aiming at being 2-5 times faster than Geant4
- Continue improvement of physics
- Full simulation and various fast simulation options
- Portable on different architectures (CPUs, GPUs and Xeon Phi's…)

Understand the limiting factors for a (10x) improvement

There will be an HEP community open meeting to review progress sponsored by HEP Software Foundation on October 27-27 at CERN
- Everybody welcome! https://indico.cern.ch/event/570876/

The initial ideas sounded easy

Transport particle
In groups (baskets)



Scheduler

Basket of tracks

Dispatching

Basket of tracks

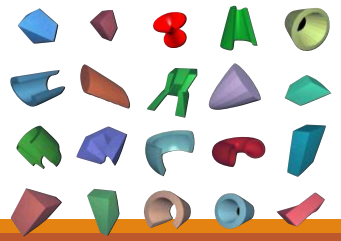
MIMD

SIMD

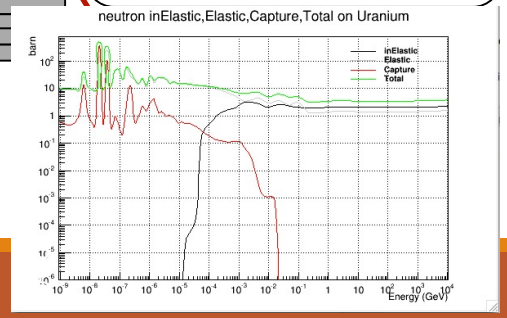
Geometry navigator

Physics

Geometry algorithms



x-sections

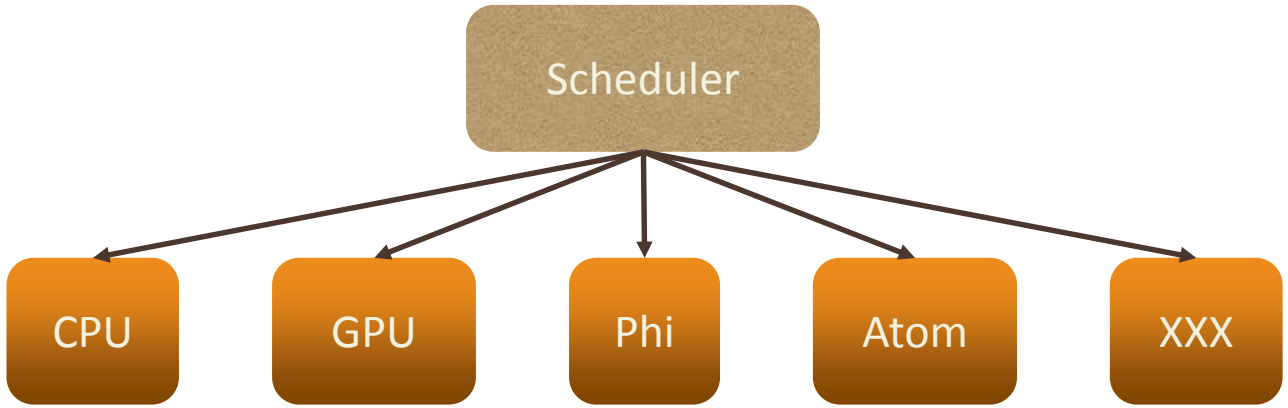


Reactions

Challenges

Overhead from reshuffling particle lists should not offset SIMD gains

Exploit the metal at its best, while maintaining portability



Test from the onset on a “large” setup (LHC-like detector)

- Toy models tell us very little – complexity is the problem

We see a substantial speedup between GEANTV and GEANT4 in similar conditions without vectorisation

- But since this is with “fake” tabulated physics, this is not yet a benchmark but a verification that rebasketization does not slow us down

Here how

"Backend" is encapsulating standard types/properties for "scalar, vector, CUDA" programming; makes information injection into template function easy

Portable HPC?

Long-term maintainability of the code => write one version of each algorithm and specialise it to the platform via template programming and low level optimised libraries

Dispatch structure of array data to functions with vector signatures

- Template on type sets, which are defined by backends
- Use backends to implement generic functions to operate on data
  - Basic arithmetic, load/store, masking operations (assigns, blending, gather/scatter), numerical limits, …
- Use backends to insulate technology/library: Vc, UME::SIMD, Cilk+, VecMic, …

A Xeon Phi specific backend (UME::SIMD) is being developed in collaboration with CERN's Openlab

```
distance( double &);
```

```
distance( vector_type  &);
```

Dual interfaces

Scalar interface

Vector interface

```
template<typename Real_v>
static void
common_distance_function( Vector3D<Real_v> input,
            Real_v &output )
{
   // Single kernel algorithm using Backend types
}
```

Kernels

Backends

```
Scalar.h
template <typename T =
Real_s>
class ScalarT {
public:
  using Real_v   = T;
  using Double_v = Double_s;
  …
}
// Functions operating with
backend types
```

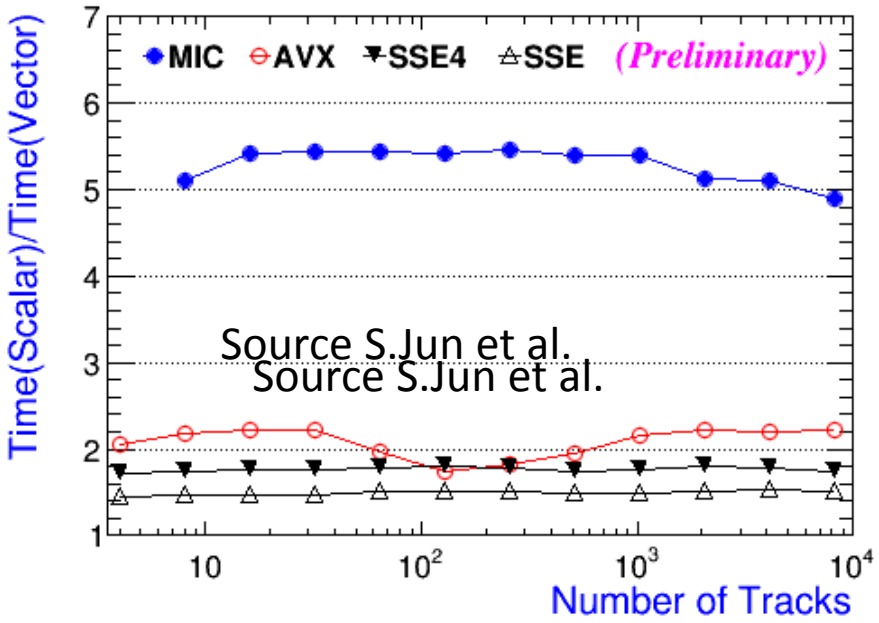
```
UMESimd.h
{
using Real_v   =
UME::SIMD::SIMDVec<Real_s,
SIMDWidth<Real_s>()>;
 using Double_v =
UME::SIMD::SIMDVec<Double_
s, SIMDWidth<Double_s>()>;
  …
};
// Functions operating with
backend types
```

```
VcSimdArray.h
{
using Real_v   =
Vc::SimdArray<Real_s, N>;
using Double_v =
Vc::SimdArray<Double_s, N>;
  …
};
// Functions operating with
backend types
```

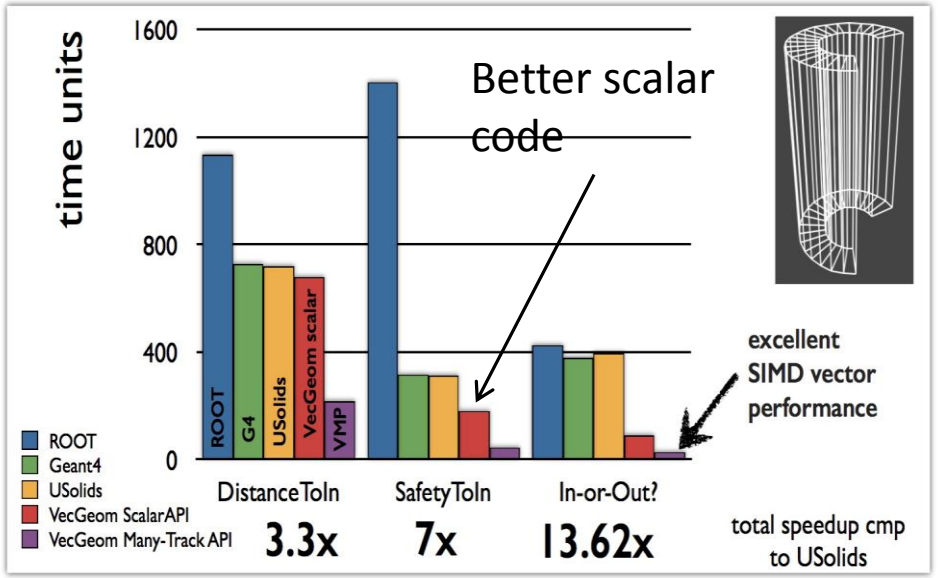
For Vc see: http://code.compeng.uni-frankfurt.de/projects/vc

Gains from vectorisation

- Vectorisation of Compton scattering shows good performance gains
- Vector code is better scalar code!



Source S.Wenzel



- Geometry is 30-40% CPU time on Geant4, (depending on the cuts)
- Substantial performance gains also in scalar mode

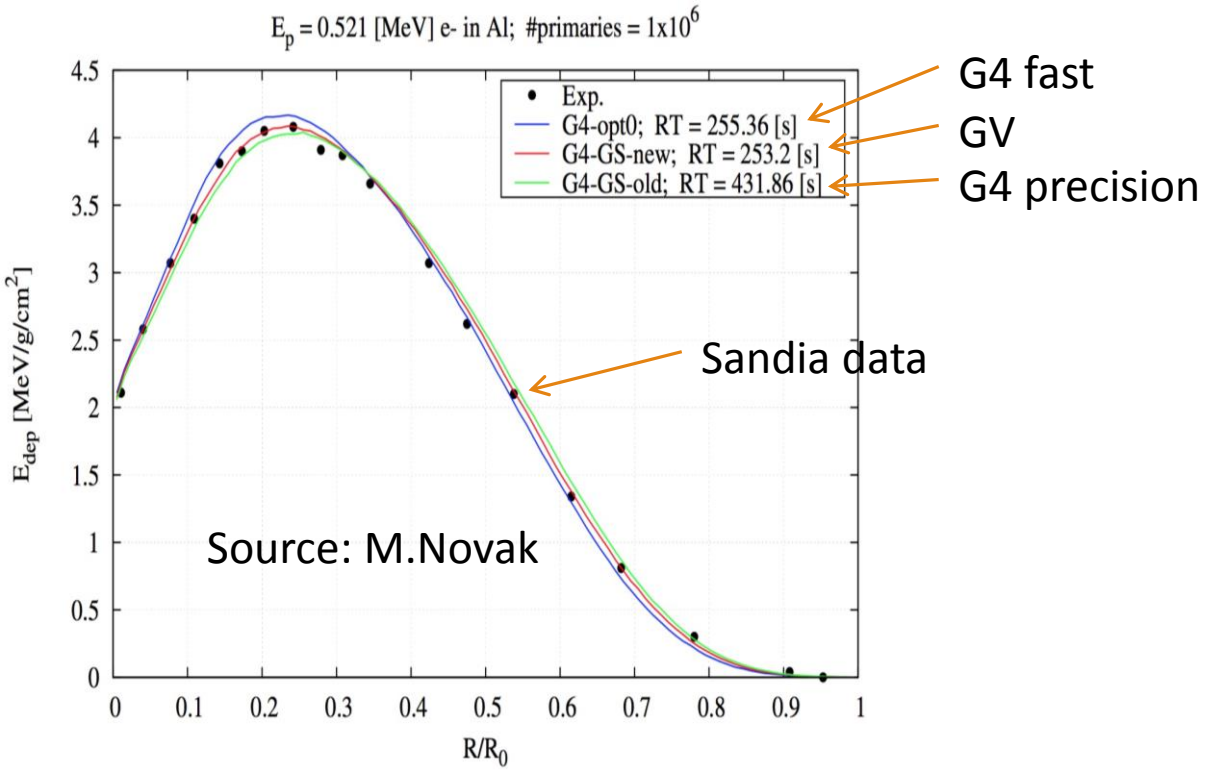
EM – a new start?

We are developing an independent EM library that can be used both from GEANTV and GEANT4

It will express the best performance with vectors when used by GEANTV

It turns out that vector code is better scalar code

A fresh look at multiple scattering has allowed to develop a better algorithm



Redesigning simulation

Maintain minimal coupling between components, in particular kernel and physics

- Keep transportation in the kernel, outside the physics

Re-use as much as possible the design of Geant4 physics

- Avoid reinventing the wheel

Streamlining by reducing the inheritance depth

- Replace whenever possible, virtual polymorphism with static (template) polymorphism
- Offer *thin* interfaces for direct calls to cross sections and final-state models

Build into the design provision for biasing and fast simulation

Redesigning simulation

Design components that can be used by different simulation codes

- Highest performance when used in vector mode
- Portable on CPUs and accelerators thanks to well identified backends (do NOT spread ifdefs in the code!)
- Make sure they can be used by GEANT4 and GEANTV at least

Progressively build a set of tests as granular as possible taking the data from a database

- Introduce automatic alarms for failing tests

Perform continuous integration and testing

- Nightly
- At every pull request

Timescales

The lifecycle of a simulation package is decades

Introducing changes requires close coordination with the customers

In the case of LHC, anything new has to be delivered before 2019 to be even considered before the end of Run 3

- Need LS2 (2019-2020) for commissioning

This requires a level of planning that is hardly compatible with the “informal” nature of the resource commitments

A look at the future

The two major players will remain GEANT's and FLUKA

- Strengthening the collaboration (e.g. on inter-comparisons would help)

As the problem grows more complex and more demanding, the two different models

- Medium size developer team with lightweight formal organization for FLUKA
- Large and formal collaboration with varying formal commitment and long term planning for GEANT4

May not be optimal

HEP (its institutions) should perhaps take a more "organic" role in the development of simulation codes

- Since there is already a very large investment in the community (CERN, IN2P3, FNAL, SLAC, KEK, ESA and few others)

A look at the future

Move from a “logical” toolkit to a more “physical” toolkit

- From “use what you need” to “download / install what you need”
- Independent libraries with defined interfaces

More independent but more cohesive working groups working around a modular library

- Better defined programme of work and single code base
- More long term “institutional” involvement in ensuring the maintenance and development

Enhanced validation

- Continuous integration
- Unit tests
- Sensitivity analysis
- Automatic alarms

A look at the future

Closer interaction with the users

- Meetings like LPCC are excellent, but they are far apart
- GEANT4 technical forum is a good example
- We should strive to cross community boundaries for MC users
- Better use of experiment high-quality data for regression testing

Better packaging and configuration control

- Automatic installers (Mac OS X, Linux, Windows)
- Tools like apt, dpkg, fink, homebrew and so on
- Modularity allowing better synergy with ROOT and experimental code

Conclusions

Simulation is essential for HEP (and beyond)

Current simulation codes respond well to the needs of the HEP community

The increase of requirements of all sort (speed, accuracy, functionality) may stress the current development and maintenance model and pose substantial technical and human challenges

A more organised development strategy could better exploit the resources that are available in the community and beyond in order to respond to the new challenges