The GeantV project: preparing the future of simulation

Federico Carminati

for the GeantV project

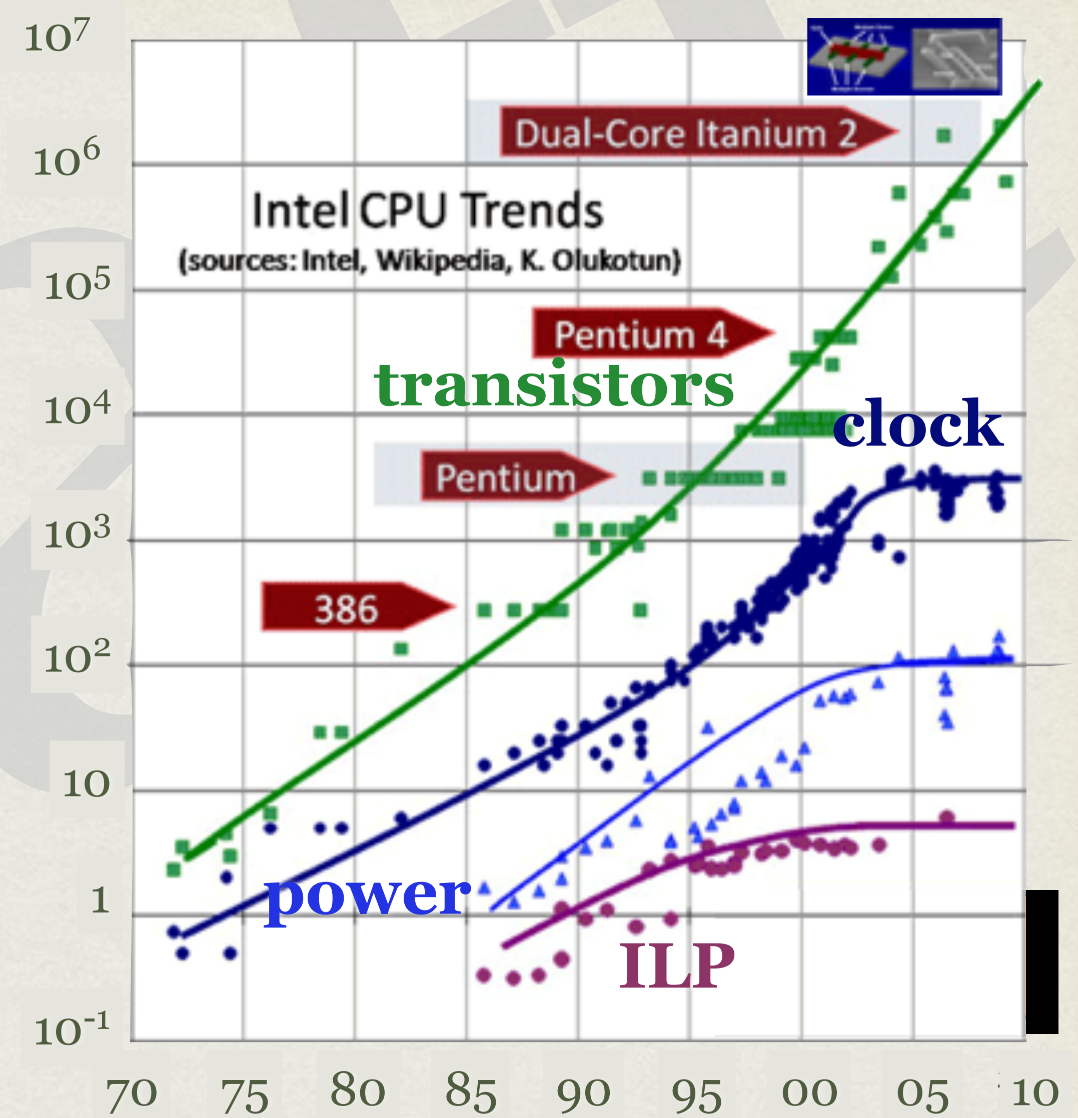
G.Amadio (UNESP), J.Apostolakis (CERN), M.Bandieramonte (Catania), A.Bhattacharyya (BARC), C.Bianchini (UNESP), R.Brun (CERN), P.Canal (FNAL), F.Carminati (CERN), L.Durhem (intel), D.Elvira (FNAL), J.de Fine Licht (CERN), A.Gheata (CERN), S.Jun (FNAL), G.Lima (FNAL), A.Mohanty (BARC), T.Nikitina (CERN), S.Novaes (UNESP), M.Novak (CERN), W.Pokorski (CERN), R.Seghal (BARC), O.Shadura (CERN), S.Vallecorsa (CERN), S.Wenzel (CERN),

SCOPE OF THIS TALK

- Update on the status of the GeantV project
- Provide *circumstantial* evidence that we are on the right track with the objectives of the project

GeantV: motivations
(even if you are *familiar* with them)

- Performance of our code scales with clock cycle (hence is stagnant!)
- Needs will increase more than tenfold and the budget will be constant at best
- Exploiting SIMD is key to achieve performance
- Portability, better physics and optimisation will be the targets
- Simulation can lead the way to show how to exploit today's CPU's resources more effectively in complex applications



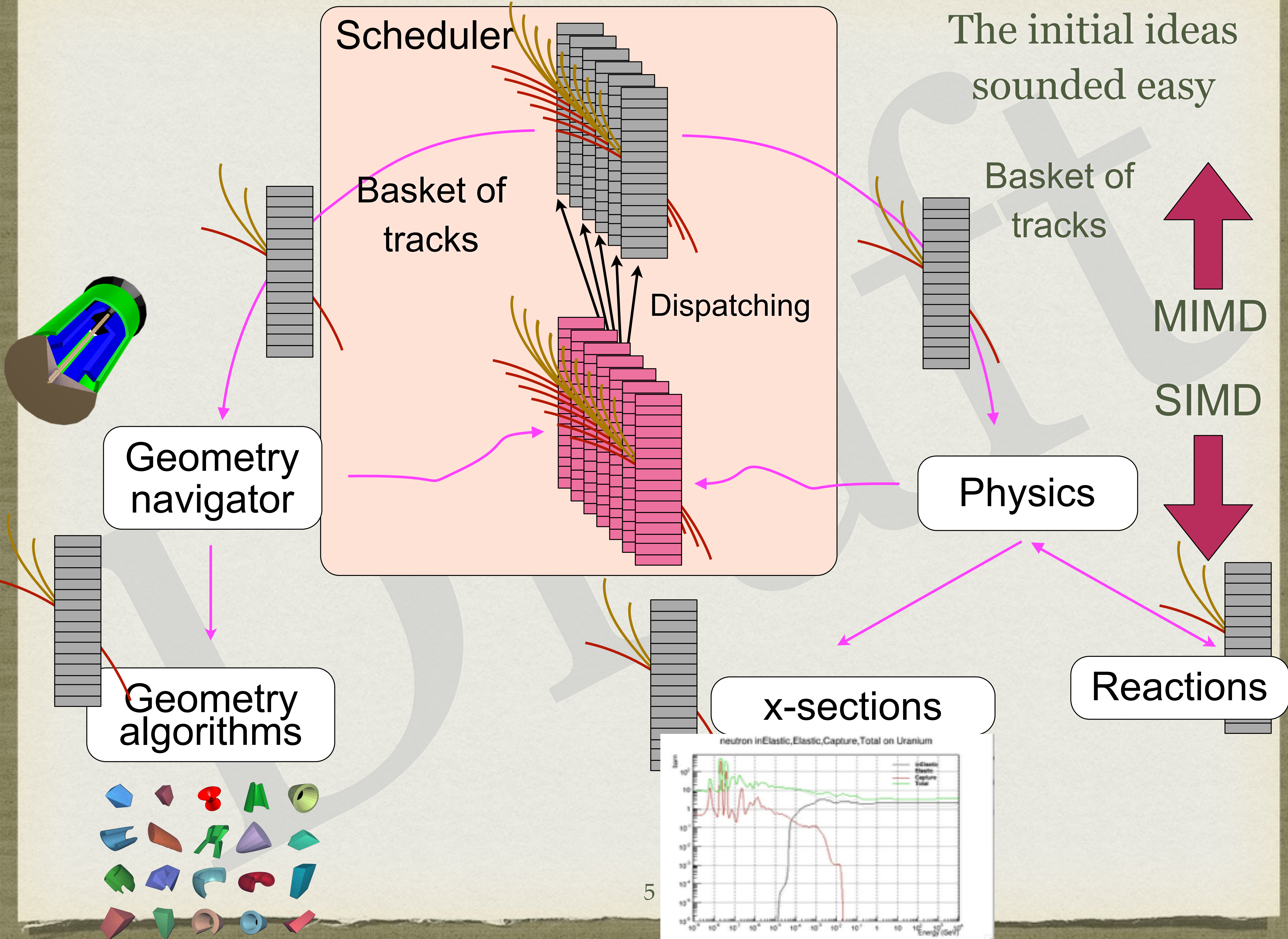
- Seeking ways to write code portable between CPU with vector units or not and accelerators (GPU, Xeon Phi)

CHEP2015
OKINAWA, japan

What do we want to do?

- Develop an all-particle transport simulation programme with
  - A performance between 2 and 5 times greater than Geant4
  - Continue improvement of physics
  - Full simulation and various options for fast simulation
  - Portable on different architectures, including accelerators (GPUs and Xeon Phi's)
- Understand the limiting factors for a one-order-of-magnitude (10x) improvement

The initial ideas sounded easy



Challenges

- The *reshuffling* of the particle lists introduces an overhead which should not offset the gains by the vectors

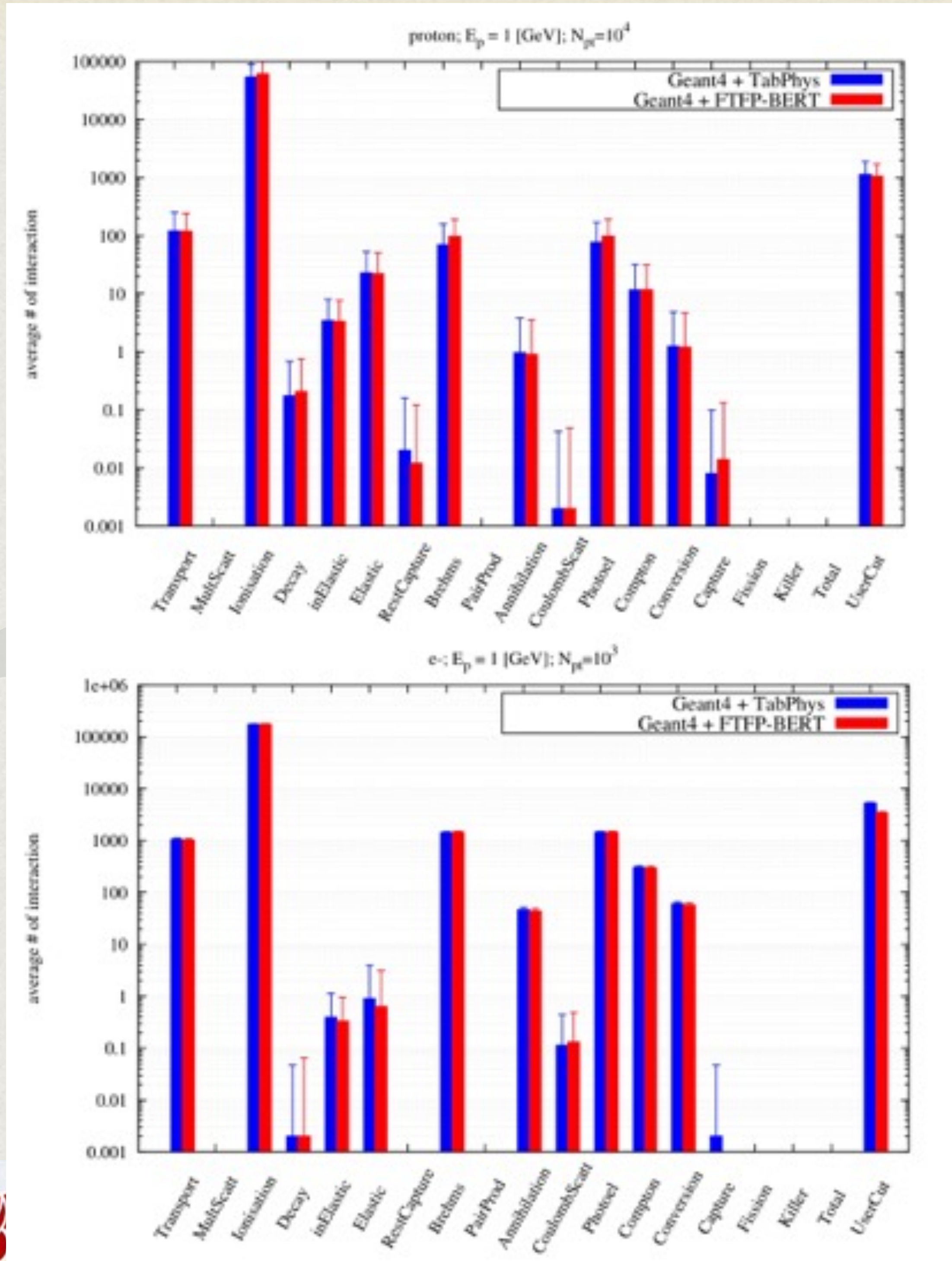
- To obtain substantial gains it is important to exploit the hardware at its best, but this, by definition lead to non portable code

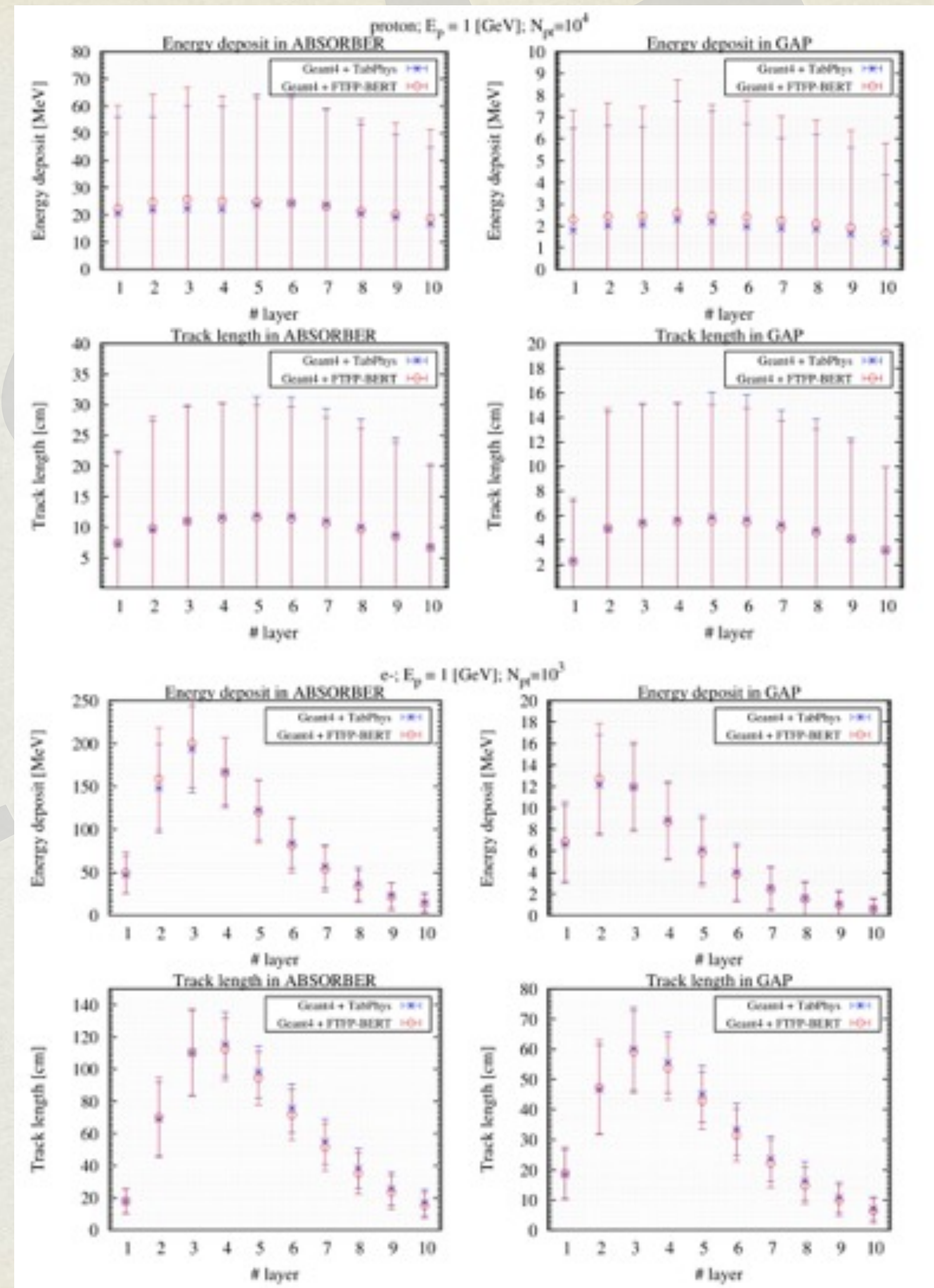
- A “small” setup will not teach us anything, we have to demonstrate speedup on a large (LHC-like) detector

What about physics?

- In order to test the prototype we need a “reasonable” shower development
- We have developed *tabulated* physics processes for GeantV
- And we have back ported them to Geant4 for verification
- Now we have a quick tool for developing realistic showers
    - Can this be developed into a fast simulation tool?

EN03 FTFP-BERT vs TABULATION

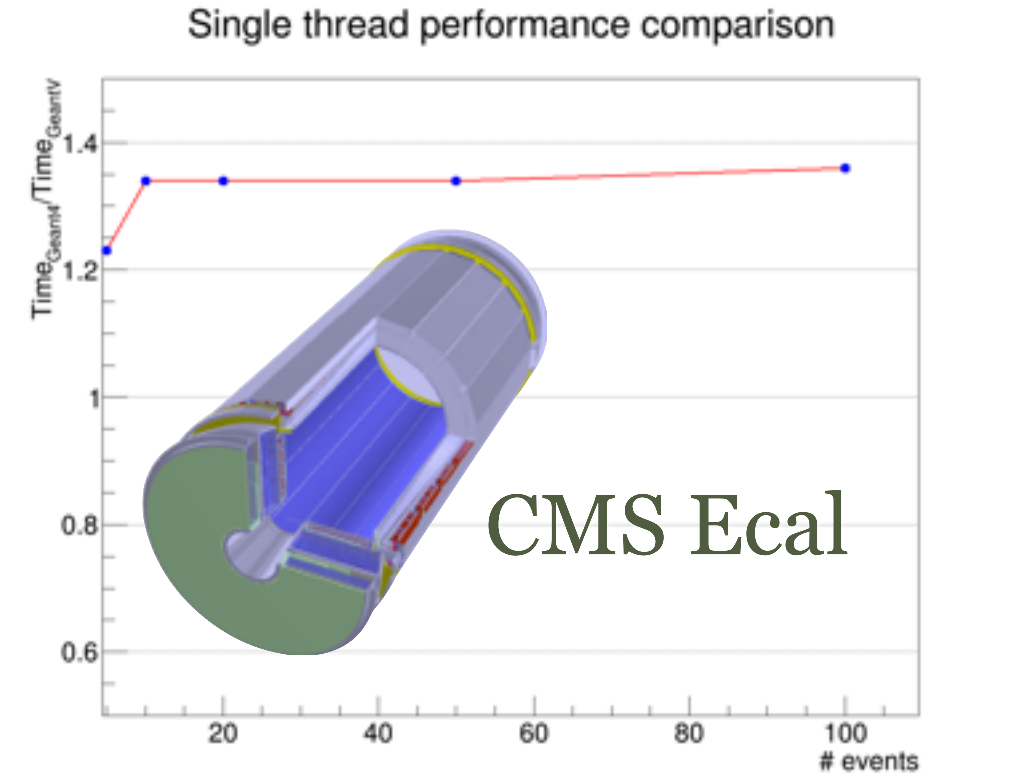


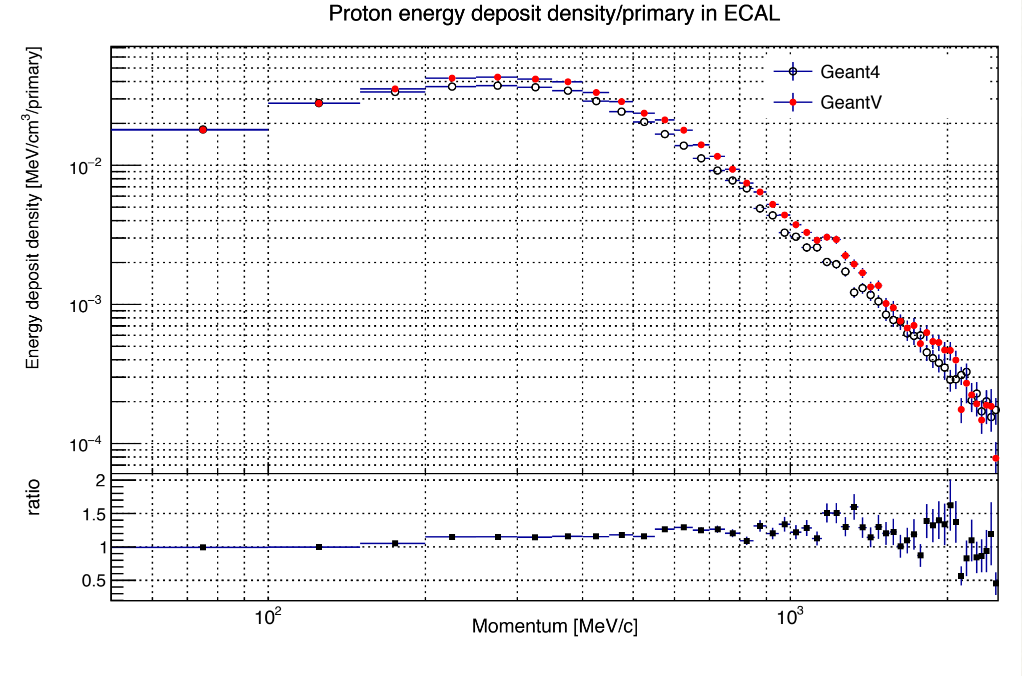


CHEP2015
OKINAWA, japan

Basket (no-)overhead

- CMS2015 GDML geometry with flat 1MeV cuts and tabulated physics
- Comparison (with tabulated physics) of
  - The multithreaded scheduler using TGeo (the geom package of ROOT)
  - Geant4
- NOT a performance comparison!!
  - Cannot compare functionality of the two programmes
- But a *circumstantial* proof that the overhead introduced by basket handling is under control
- What still has to be improved is the multithread scalability





GEOMETRY

- We have developed a library of vectorised geometry algorithms to take maximum advantage of SIMD architectures
- See VegGeom poster presented by S.Wenzel
- We obtain excellent performance gains also in scalar mode

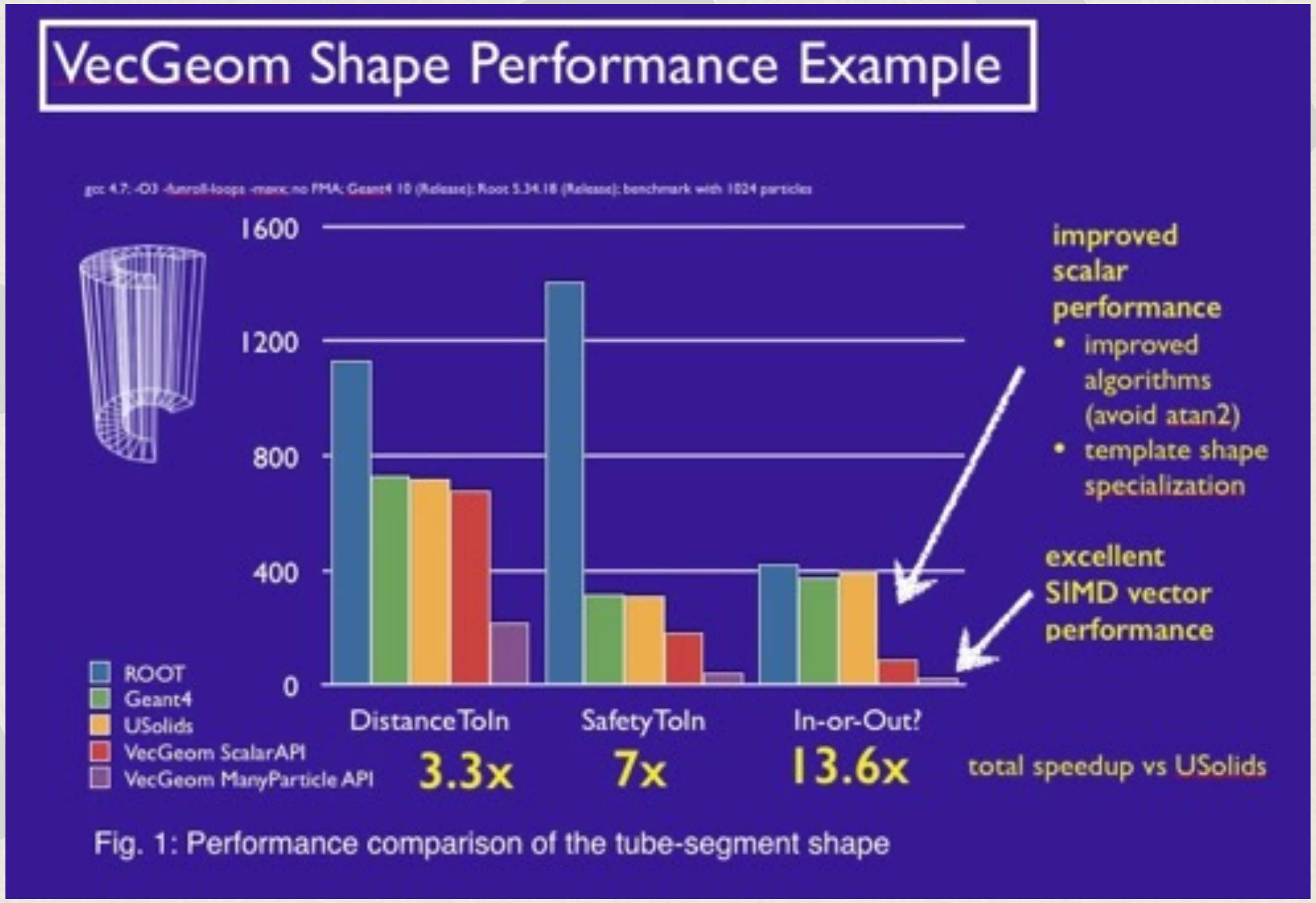


Fig. 1: Performance comparison of the tube-segment shape

The code is available in the AIDA usolid library and is being validated for Geant4 use

CHEP2015 OKINAWA, japan

PORTABILITY

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

- To ensure long-term maintainability of the code we decided to write one single version of each algorithm and to specialise it according to the platform via template programming and low level optimised libraries (VC in our case)
- Results are quite encouraging: may be portable HPC is NOT an oxymoron after all…

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

1 particle API → Common C++ template functions ← Many particle API (SIMD)

double distance(double);

Vc::double_v distance(Vc::double_v);

```
template<class Backend>
Backend::double_t
common_distance_function( Backend::double_t input )
{
   // Algorithm using Backend types
}
```

```
struct ScalarBackend
{
   typedef double double_t;
   typedef bool   bool_t;
   static const bool IsScalar=true;
   static const bool IsSIMD=false;
};
```

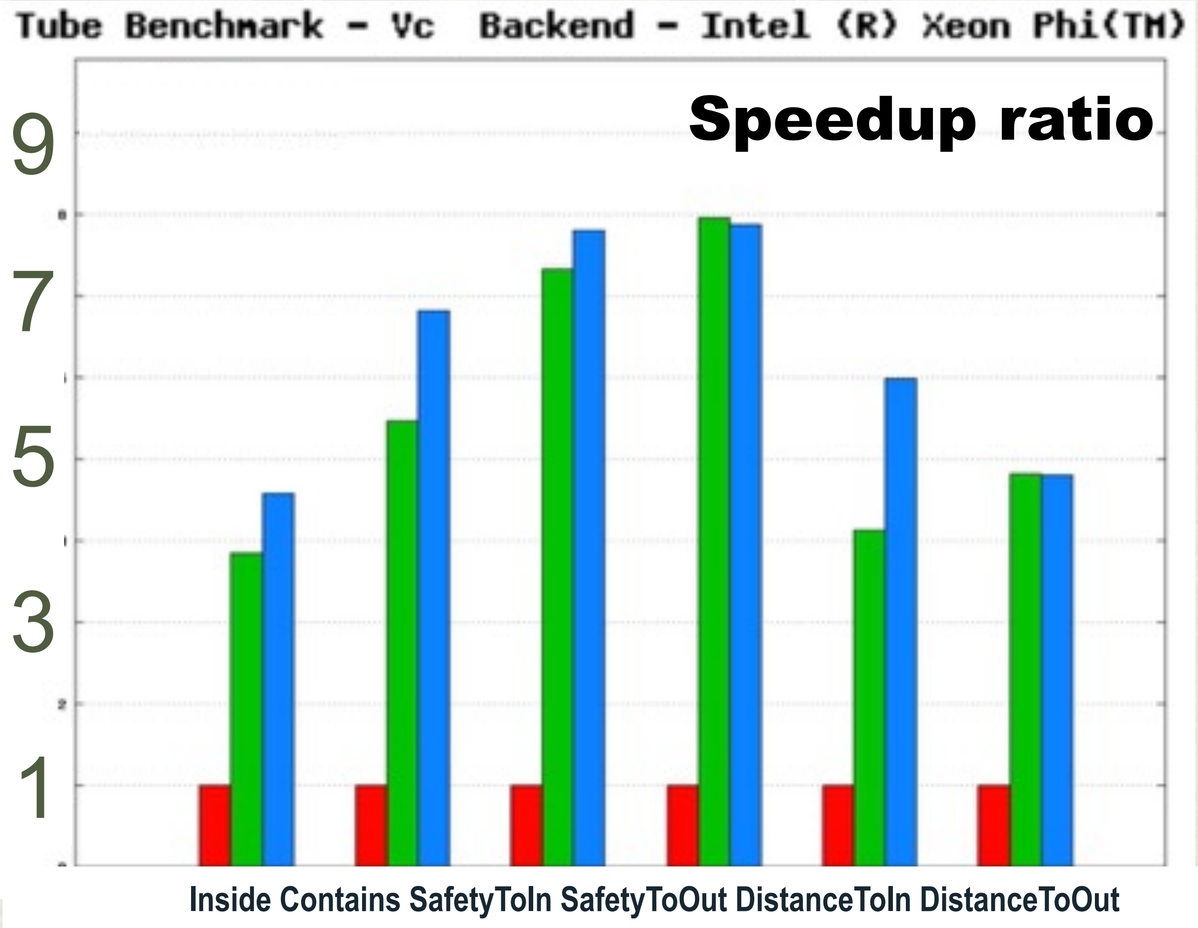
```
struct VectorBackend
{
   typedef Vc::double_v double_t;
   typedef Vc::double_m bool_t;
   static const boolIsScalar=false;
   static const bool IsSIMD=true;
};
```

CHEP2015
OKINAWA, japan

How does it work?

Remember: No Change In The Algorithm!

- Results obtained by CERN and UNESP Intel IPCCs on Xeon Phi
- SIMD optimisation gives us more than half order of magnitude
  - Remember: for single thread SIMD the max lim is 8 (IMCI) – difficult to do better…



https://software.intel.com/en-us/articles/ipcc-at-cern-european-organisation-for-nuclear-research

How does it work?

Remember: No Change In The Algorithm!

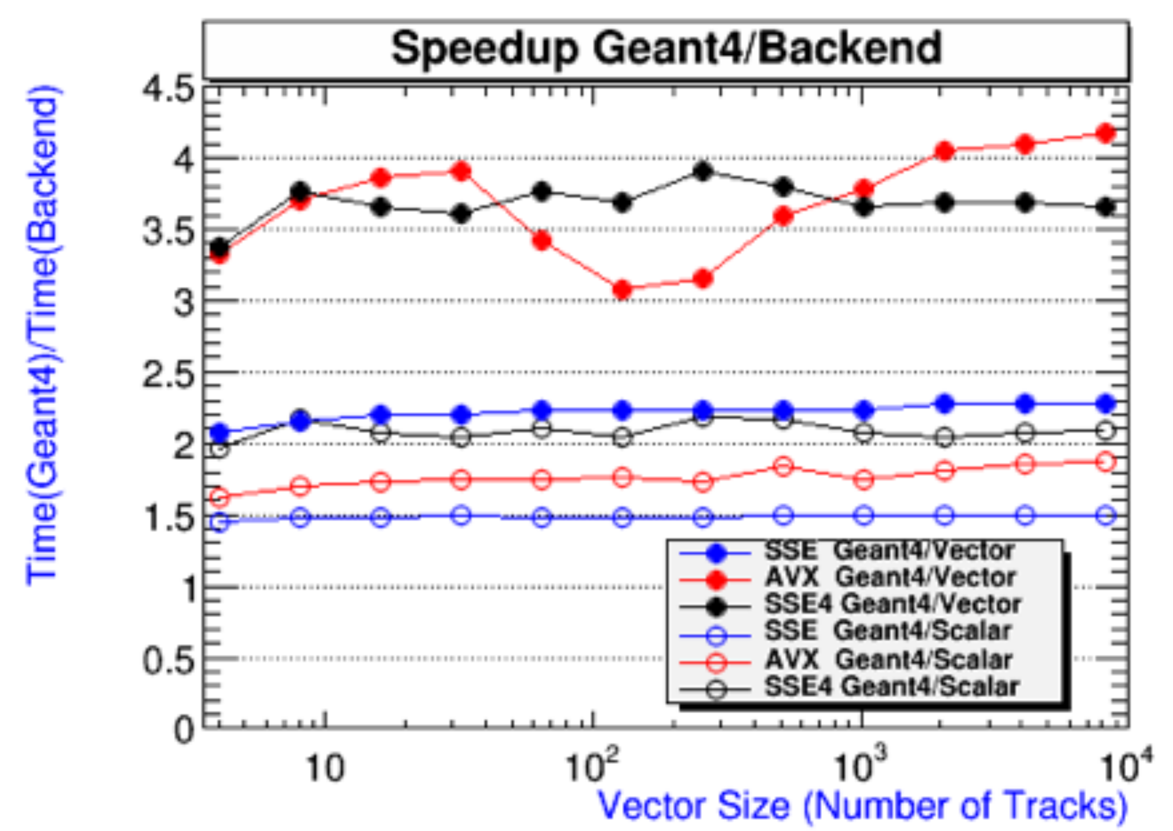
- Results obtained by the FNAL group on NVIDIA GPUs

Here the results from Philippe on GPUs

CHEP2015
OKINAWA, japan

What about physics?

- Physics vectorisation has started with the electromagnetic processes
- A vectorised version of the Compton scattering has been developed showing good performance gains
- Again, vector code is better scalar code!
  - For more see P.Canal talk *Detector Simulation On Modern Processors*
- We will consider whether to retrofit this into Geant4



Courtesy of S.Jun & J.Apostolakis

CHEP2015 OKINAWA, japan

Restating our case

- We have developed three main components
    1. A multithread scheduler to handle the particle baskets
    2. A vectorised geometry library and navigator
    3. A vectorised example of physics model for Compton scattering
- Our results indicate that
    - Basket handling introduces a minimal overhead
    - SIMD optimisation can gain around half an order of magnitude in performance
- We believe we are on track with our objective
- A success oriented interpretation of our results till no could be 1.4 (scheduler) x 4 (physics and geometry) ~ 6,
- Of course the proof of the cake will be in the eating

CHEP2015
OKINAWA, japan

Next steps

- Performance tests using the scheduler and VecGeom navigation on large geometry (CMS2015 but also some other large detector)
- Repeat the test with the introduction of
    - Voxelisation in VecGeom
    - Vectorised Compton
    - Vectorised transport in Mag Field
- This will allow us to have a reasonable performance assessment
- Develop simple classes for materials and particles to be able to run on coprocessors to enable the GPU and Xeon Phi full benchmark
- This should be done by fall 2015

Conclusions

- Work toward a credible demonstrator for a high performance particle transport code is progressing steadily
- The techniques developed have the potential to benefit to many other codes within and outside of HEP