GeantV Conclusions and Follow-up

Pere Mato for the GeantV team

Date

GeantV final publication

- To finalise the GeantV prototype exercise we are writing a technical paper with all the design concepts, implementation choices of the prototype, the evaluation of its performance and all the findings and lessons learnt
  - The content will correspond to the results presented today
  - It will be submitted to Computing and Software for Big Science (Springer)
  - The plan is to submit it before the end of the year

Lessons learnt

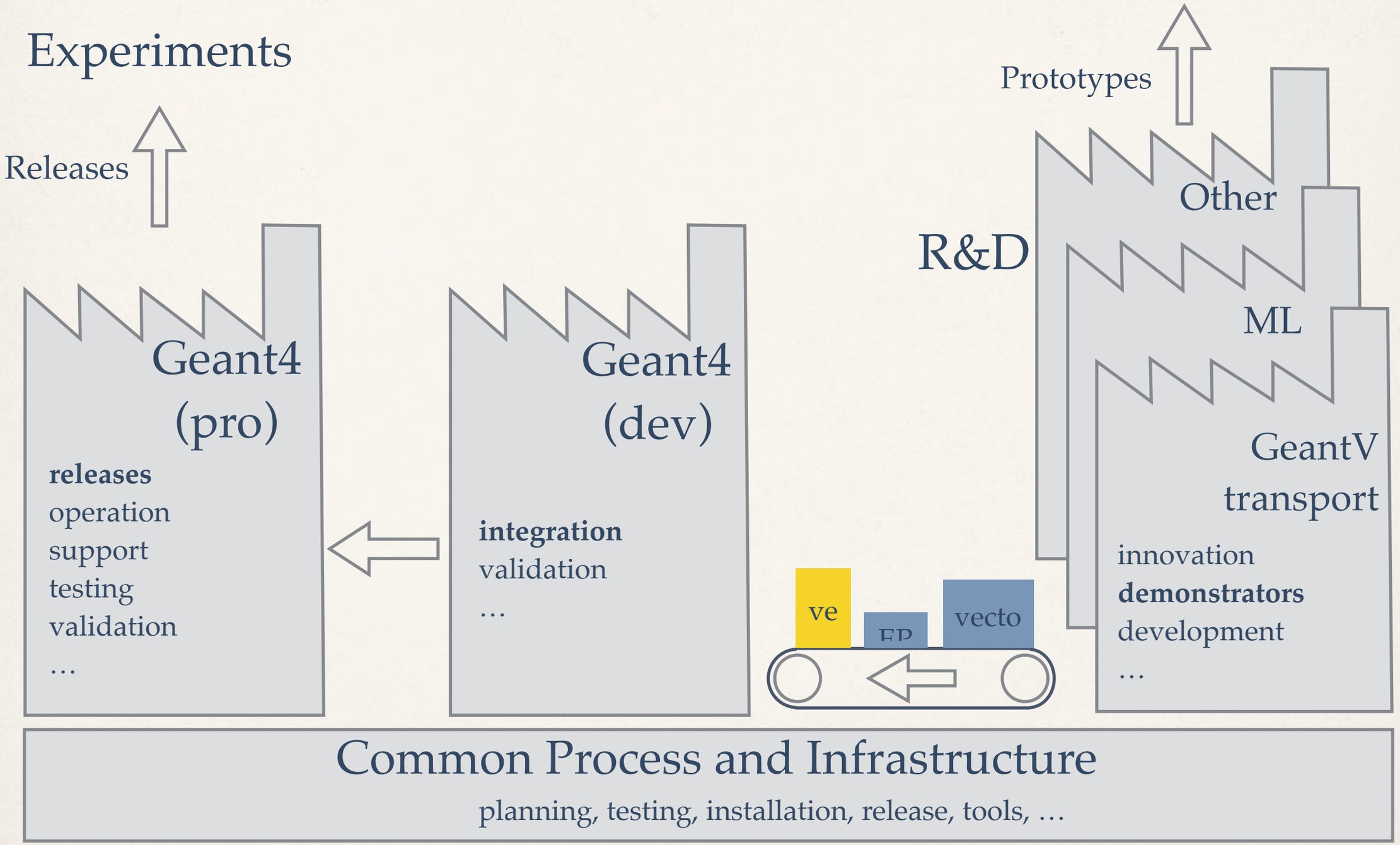
- Libraries developed for the GeantV prototype have resulted to be very useful (e.g. VecCore, VecGeom, VecMath)
  - successfully integrated in Geant4, ROOT, etc.
- Re-writing and modernising large parts of Geant4 potentially could bring us a factor of 2.0 ± 0.5 in performance (depending on the CPU/caches)
  - compact code, better data formats, data locality, less virtual functions, etc.
- Vectorisation (organising the work in baskets of particles) does not bring the expected speedups. In some cases deteriorates the overall performance.
  - large overheads in continuously reshuffling particles in baskets, dealing with the processing tails
- The initial work (and very partial) with GPUs demonstrated to be very impractical
  - same problems as for vectorisation, but with larger baskets to make it efficient, copying data in and out of device, code duplication in and out, etc.

Where Do We Go From Here?

Current Simulation Strategy

- Ensure the support of experiments through **continuous maintenance, support and improvement** of the Geant4 simulation toolkit
- Ensure that new developments resulting from any R&D program **can be integrated and deployed in Geant4** once they are demonstrated to be beneficial
  - Not afraid of major changes (most probable with a backward incompatible API) in the structure as long as the benefits are sizable
- Rely on the Geant4 collaboration to evolve the toolkit by integrating the R&D successful results, and be ready to support the experiments for migrations that will require major changes to the user code
  - Create Geant4 Task-Force for R&D (see slide later)

Simulation Strategy



Geant4 R&D Task Force

- **Promote and survey research activities** on potential software architectural revisions and the exploitation of emerging technologies or computing architectures that would be beneficial to Geant4
- **Ensure the visibility** of these R&D activities inside and outside of collaboration
- Where appropriate, conduct benchmarking comparison and **provide/assist communication and support among the R&D activities**
- Make timely **assessment reports to the Steering Board** with solid proof of benefits and eventually spawn dedicated task forces to create workflow, estimate required resources and drive that particular development for integration into the code base

Three Main Axis of Development

- **Improve, optimise and modernise the existing Geant4 code to gain in performance for the detailed simulation**
  - Re-structure the code to make possible major changes (task-oriented concurrency, specialisation of the physicists, better data formats, etc.)
  - Some recent successes but we need to do much more
- **Trade precision for performance using fast simulation techniques** both with parameterisations or with ML methods and integrate them seemly in Geant4
  - Use detailed simulation to 'train networks' or to 'fit parameters' that later can deliver approximative detector responses well integrated within Geant4
- **Investigate the use of 'accelerators'** with a different approach than accumulating particles in baskets and offload the processing

Optimise and Modernise Geant4

- We have already plans for a number of actions. Some examples:
  - Integrate the VecGeom navigator in Geant4
  - Geometry specialisations (perhaps with some code generation)
  - Group the state of ‘particles’ in order enable batch processing of several particles
  - Physics code specialisation of gamma/electrons inside an homogenous volume (implementing shortcuts for the generic framework)
  - Concurrency with a task-oriented programming model
- Development of the tooling and expertise for producing and analysing performance profiles
  - Essential for optimising the code

Fast Simulation Techniques

- Developing a customisable tool for data production in a regular calorimeter-like detector for training and fitting model purposes
  - Input particle energy spectrum, output histograms with different observables
  - Essential for validating the different techniques
- Evaluating several generative ML models:
  - Auto-Regressive models, Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs)
- Revision of the GFlash implementation
  - Specific tuning for materials and geometry is required. Working on automating the recipe
- Prototyping occurrence biasing of the charged particles

Investigate the use of ‘accelerators’

- Goal: transform the very heterogenous Geant4 HEP particle transportation into a more homogenous computational problem
  - explore ideas of transforming complex geometries in simple ones
  - identify and isolate few physics processes that accounts for most of the computational time
  - organise the computational work to minimise copying data to and from device
- Development of representative ‘demonstrators’
  - before embarking in a massive conversion and re-engineering of large parts of the code we need to demonstrate its feasibility
- Study sustainability aspects of the code and the use of portability libraries
  - gain experience in using libraries such as Kokkos, Alpaka, etc.

Discussion