CmsToyGV
a first GeantV prototype for CMS

G.Lima (FNAL), A.Gheata (CERN)
for the GeantV team

CMS Simulation Meeting
CMS Week, Feb. 06, 2018

GeantV and CMS

- GeantV brings many new developments
  - a new transport engine prototype, developed and supported by a partnership of CERN, Fermilab, BARC/India, CIC/IPN/Mexico and financial support from Intel.
  - our preliminary results have demonstrated significant computing performance gains benefitting from fine-grained parallelism using SIMD vectorization
  - our team is motivated and ready to support CMS in moving forward with our released products, and committed to provide all necessary support
  - Geant4+VecGeom is a very good start. Performance gains due to improved data structure and algorithms. SIMD vectorization gains require the new GeantV transportation engine
  - The final goal is to integrate CMSSW and GeantV, but how to get there?

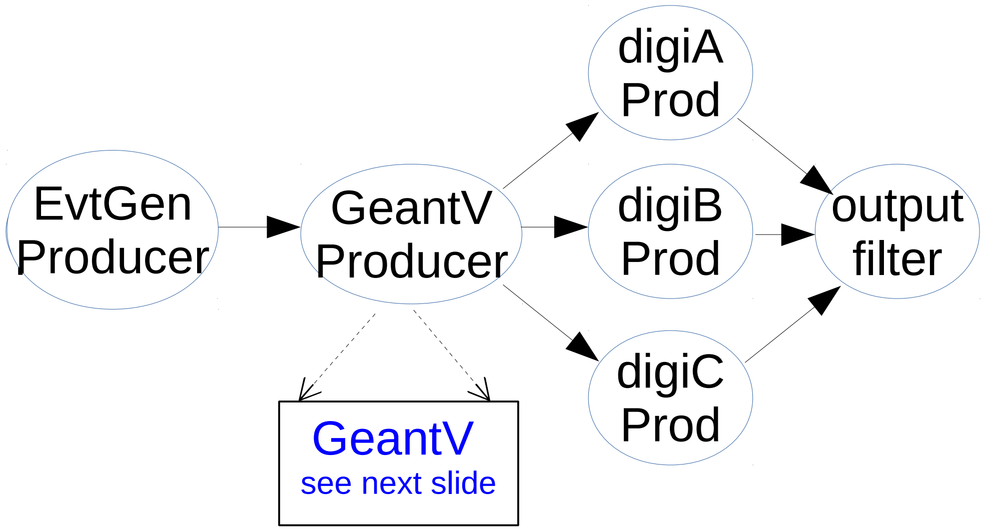
- From CmsToyGV to CMSSW+GeantV
  - **first step:** integrate GeantV into existing package toy-mt-framework, by Chris Jones *et. al.*
  - goal is to develop and adjust the GeantV engine interfaces to a simple framework, to allow the external framework to control the simulation event loop, and the initial configuration of the simulation engine

    → a first application where the GeantV engine is tested in realistic conditions.
  - **second step:** CMS simulation group will integrate GeantV into CMSSW, with support from the GeantV team
  - The collaboration between GeantV team with CMS will help identify and resolve technical issues, promoting faster development cycles

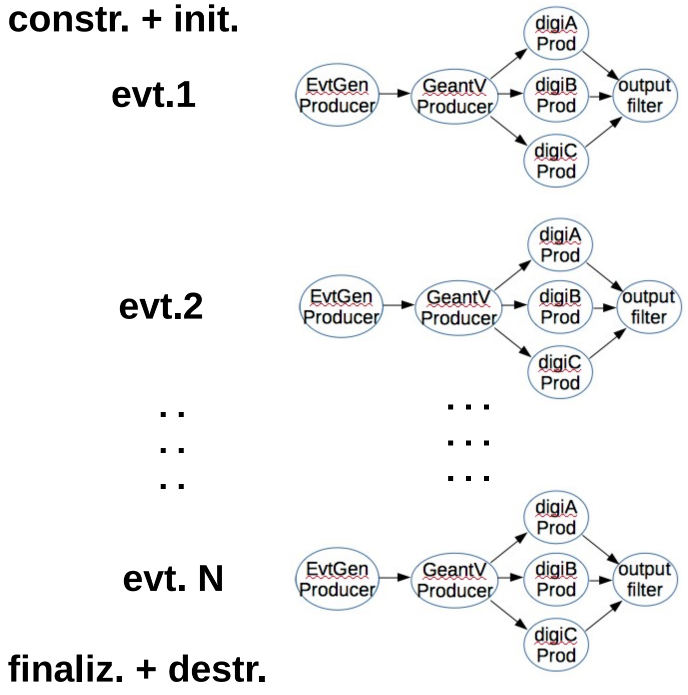
CMS side: the toy-mt-framework

- The toy-mt-framework package (without GeantV yet)
  - available from github, a bare-bones framework emulating the CMSSW multi-threading infrastructure (dummy tasks, no real data processing)
  - all threads processing the same event likely to run on same core
  - configuration: driven by a .json structure, easy to modify and maintain

Task-based event processing in toy-mt-framework package



Multiple events can run concurrently, according to .json configuration



GeantV transport engine

- Multi-particle interfaces: track-level parallelism
  - main source of performance gains (through SIMD vectorization)
  - algorithms interfaces receive baskets of particles → multiple particles processed in parallel
  - basketizer (new concept):
    - fills baskets (buffers) with similar tracks to maximize SIMD synchronization
    - tracks are selected according to particle type and geometry criteria
    - tracks in a basket can be from different events!
- Other GeantV features
  - generic code based on templated types and functions, promotes use of SIMD vectorization for different processors/architectures and vectorization libraries
  - redesigned code and data structures: improved memory locality, reduced cache misses
  - adaptive scheduler monitors MT performance, may adjust parallelization parameters dynamically at run time
- Vectorized components:
  - geometry algos: inside/outside, safety to surface (no direction), distance to surface (w/direction), normals
  - navigation on geometry hierarchy
  - propagation in magnetic field – uniform field for now, generic mag field expected “soon”
  - physics: only EM process available for now (scalar mode)

    physics algorithm vectorization efforst are just starting (very challenging!)

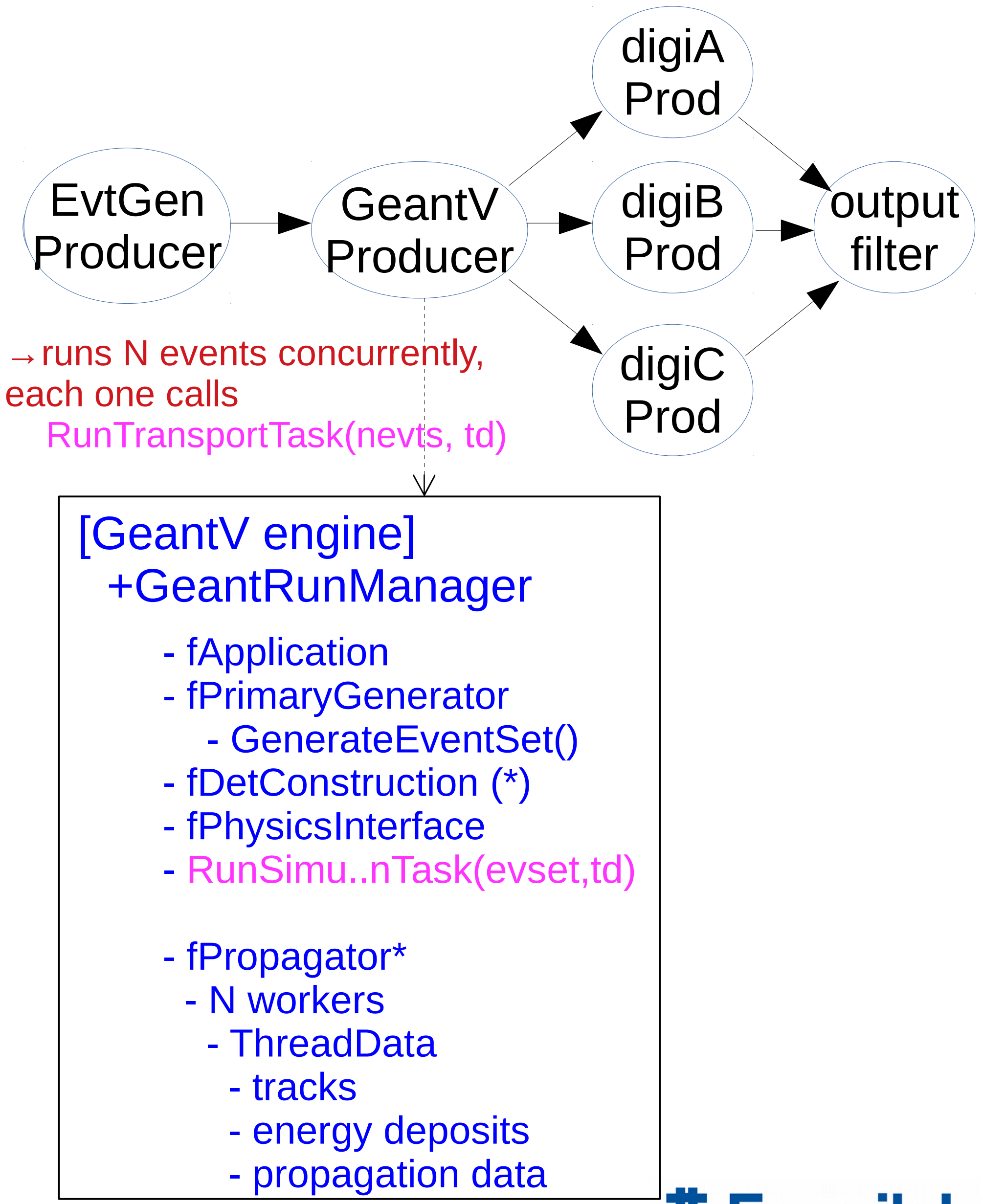
GeantV transport engine

- Components and current status
  - VecCore: new types and function implementations
    - provide a backend interface to specific vectorization libraries (currently: Vc, UME::SIMD)
    - vectorized utilities: pRNG, vectorized standard math functions and constants, etc.
    - main functionality is ready, in maintenance mode, more features to come
  - VecGeom: vectorized geometry and navigation library
    - multi-particle interfaces, while keeping drag-and-drop compatibility with Geant4
    - first vectorization results: sizable gains in unit tests, to be harvested in integrated environments
    - first production release, thanks to CMS for early adoption! Room for optimizations.
  - GeantV: the new transportation engine
    - new scheduler, *basketizer*, external interfaces for experiment's framework
    - physics (EM, HAD), processes, models, tracking, mag field, MC truth, I/O etc.
    - alpha release is ~out, beta release by 2019

Integrating GeantV: the new CmsToyGV

- GeantVProducer
  - CMS module that instantiates and fully configures GeantV
  - CMS event loop will run GeantVProducer::produce() once per event, **on concurrent threads**, which will allow GeantV to use those thread for simulations.
- GeantRunManager
  - configures and steers all GeantV components
  - provides the external interface to be called by the framework, passing events to be processed
- Each transport task will
  - get new events from CMSPrimaryGenerator and save it into thread-local buffers ThreadData *td
  - call runMgr.RunSimulationTask(evset, td)
  - Threads work **cooperatively** to complete eventSets, including processing each other's tracks (from different event sets)

Task-based event processing in CmsToyGV framework



[GeantV engine]
+GeantRunManager
- fApplication
- fPrimaryGenerator
  - GenerateEventSet()
- fDetConstruction (*)
- fPhysicsInterface
- RunSimu..nTask(evset,td)

- fPropagator*
  - N workers
    - ThreadData
      - tracks
      - energy deposits
      - propagation data

So what does CmsToyGV currently do?

- We are using CMS2018 geometry with an uniform 4T magnetic field for development and tests (propagation in generic magnetic fields will be released soon).
- At this point, only EM physics processes available in GeantV, but those are good for detailed performance comparisons with Geant4. Hadronic processes can be used, for tests (not recommended), but in this case both EM and HAD will be based on x-section tables.
- The toy framework "produces" events (from particle guns or HepMC files), converts them to GeantV format (GeantEventSet).
- The framework also controls the event loop and passes one event at a time to GeantV through each call to GeantRunManager::RunSimulationTask(eventSet, threadBuffer).

  → Concurrent CMS modules can pass one thread each to GeantV control, and GeantV will run basketizers and propagators as needed until all tracks are processed.
- GeantV receives the events, sorts the tracks through baskets according to particle type and/or volume being traversed.

  → Note that baskets can contain tracks from multiple events!
- All secondary tracks produced are basketized, and then scheduled for processing.
- Once all tracks are processed, the event is reassembled, so all produced hits and MC truth information get properly stored. Control is then returned to CMS module GeantVProducer
- GeantVProducer could append simulated information into the CMS event structure, before passing on the control to next CMS task (digitization modules).
  - Currently, other than GeantVProducer, all other CmsToyGV modules are just placeholders, but get CPU time assigned to them at the right times (Cms framework responsibility)

CmsToyGV prospects

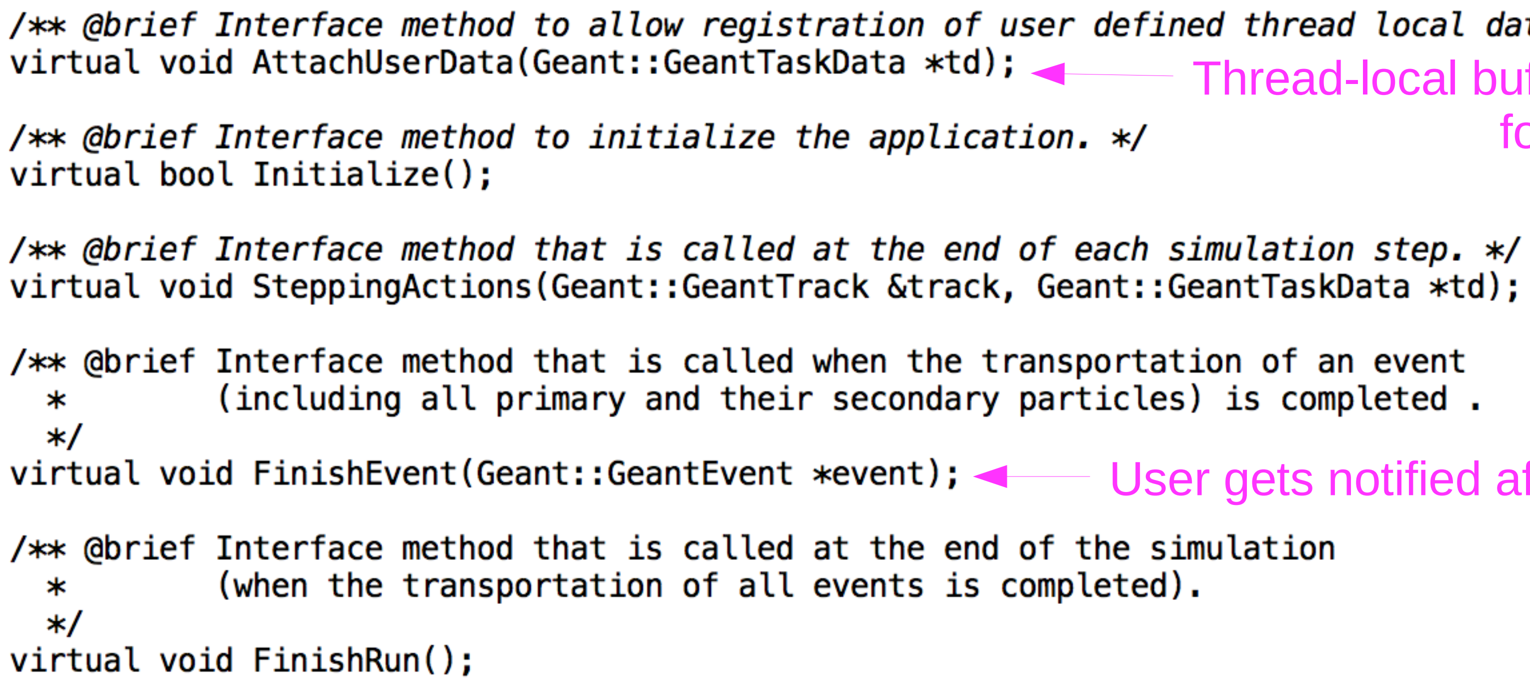
- First version of GeantV user interface for external control is available
  - interface includes a reentrant function RunSimulationTask(eventSet, threadBuffer), which provides a thread for GeantV use, and returns to external framework when events are ready
- GeantV has been interfaced with a multi-threading prototype of CMSSW
  - combined executable available as a GeantV example, from GeantV repository, at subdirectory examplesRP/GeantV/cmsToyGV (thanks to Chris Jones et.al.)
- Still lots of room for improvement
  - Detector geometry can be provided programmatically, but also in either .root or .gdml format
  - CMSParticleGun ready, HepMC input format will be available soon
  - EM Physics models fully available
    - can be used for comparisons with Geant4 (first preliminary results expected soon)
    - vectorization recently started – it is a long process
- Next milestone: port to CMSSW
  - Kevin Pedro has been contributing, and planning to work on the port to CMSSW
- Full CmsToyGV is part of the GeantV alpha release

Thanks to A.Gheata, Ph.Canal, S.Y.Jun, and K.Pedro, C.Jones (CMS)
for the help and discussions during the development of this work

Backup slides

User hooks to GeantV engine

- Class CMSFullApp : Geant::GeantVApplication

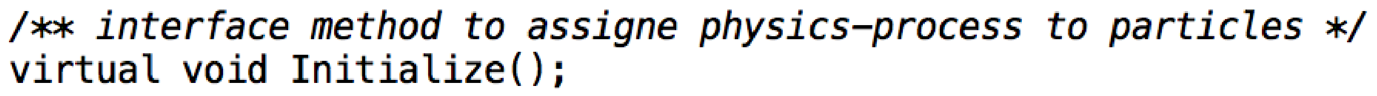


```
/** @brief Interface method to allow registration of user defined thread local data. */
virtual void AttachUserData(Geant::GeantTaskData *td);

/** @brief Interface method to initialize the application. */
virtual bool Initialize();

/** @brief Interface method that is called at the end of each simulation step. */
virtual void SteppingActions(Geant::GeantTrack &track, Geant::GeantTaskData *td);

/** @brief Interface method that is called when the transportation of an event
  *        (including all primary and their secondary particles) is completed .
  */
virtual void FinishEvent(Geant::GeantEvent *event);

/** @brief Interface method that is called at the end of the simulation
  *        (when the transportation of all events is completed).
  */
virtual void FinishRun();
```

Thread-local buffers assigned for GeantV use

User gets notified after each event is ready

- Class CMSPhysicsList : Geant::PhysicsList



```
/** interface method to assigne physics-process to particles */
virtual void Initialize();
```

User hooks to GeantV engine

- Class CMSDetConstruction : Geant::DetectorConstruction

```
/** @brief Creation of geometry (mandatory) */
void CreateGeometry() final { LoadGeometry(fGeomFileName.c_str()); }
```

.root, .gdml accepted

- Class CMSParticleGun : Geant::PrimaryGenerator

```
/**
 * @brief Pure virtual function of initialization of primary generator
 * @details Set one GeantTrack primary track properties
 */
virtual void InitPrimaryGenerator() = 0;

/** @brief  Pure virtual function that produce next event
  *
  * @param td thread local data pointer
  */
virtual GeantEventInfo NextEvent(Geant::GeantTaskData* td) = 0;

/**
 * @brief Pure virtual function that returns track
 *
 * @param n Track index
 * @param gtrack track
 * @param td thread local data pointer
 */
virtual void GetTrack(int n, Geant::GeantTrack &gtrack, Geant::GeantTaskData* td) = 0;
```

GeantEventInfo:
ntracks, pvx, pvy, pvz