ADAPTATIVE TRACK SCHEDULING TO OPTIMIZE CONCURRENCY AND VECTORIZATION IN GEANTV

J Apostolakis, M Bandieramonte, G Bitzes, R Brun, P Canal, F Carminati, J C De Fine Licht, L Duhem, V D Elvira, A Gheata, S Jun, G Lima, M Novak, R Sehgal, O Shadura, S Wenzel

ACAT 2014

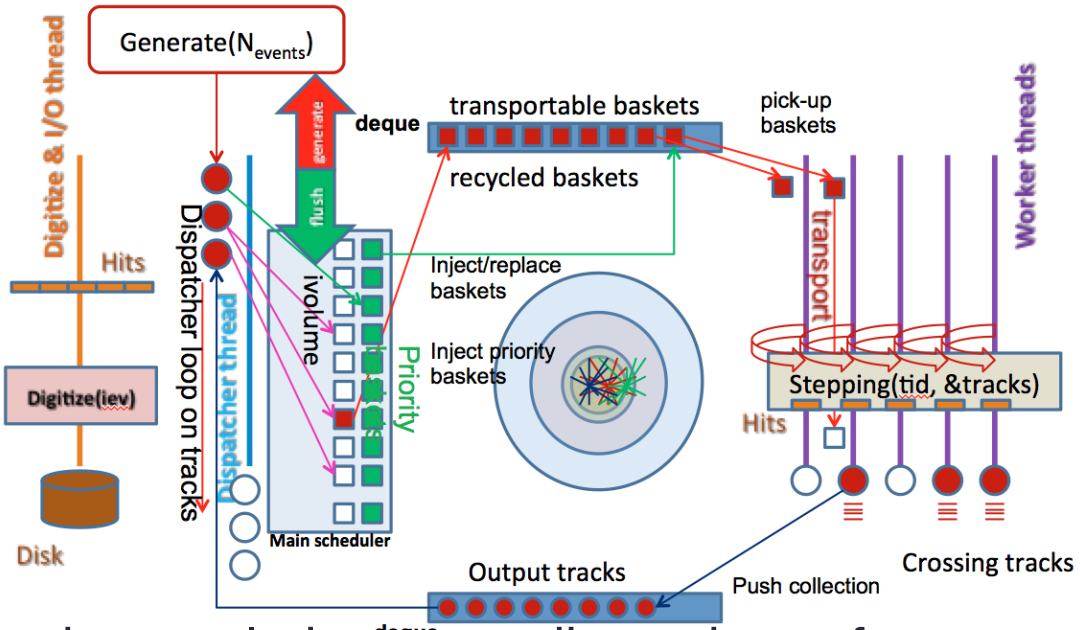
Prague, 1-5 September 2014

Outlook

- GeantV: project description
- The data model: vectors and baskets
- The track dispatching model
- Vectorization overheads
- Scalability and performance
- Optimizing concurrency
- Optimizing the model parameters

The project goals

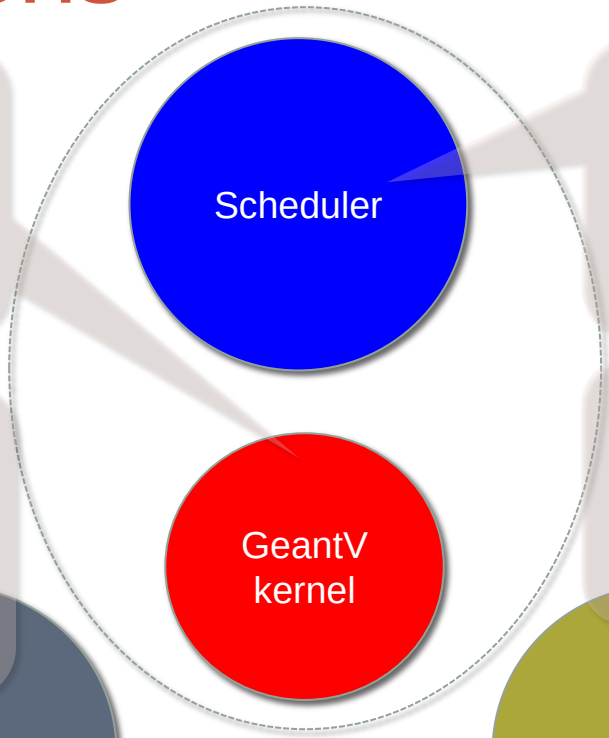
- Started in 2012
  - Prototype a new approach to speedup particle transport simulation
- Most simulation time spent in few percent of volumes
  - Enforce locality and vectorization transporting groups of tracks
  - Add parallelism on top
  - Add new entity to control the workflow



- Evolved into an ambitious project exploring many dimensions of performance
  - Locality (cache coherence, data structures)
  - Parallelism (multi/many core, SIMD)
  - Vector dispatching down to algorithms
  - Transparent usage of resources (CPU/GPU)
  - Algorithm template specializations & generality of code (next talk)
  - Physics & geometry algorithm improvements

R&D directions

- Data structures, SOA types
- Concurrency libraries
- Steering code
- Base classes, interfaces
- Management and configuration
- Testing, benchmarking, development tools



- Locality by geometry or physics
- Dispatch efficient vectors
- Manage concurrency and resources
- Schedule transportation, user code, I/O based on queues
- Optimize model parameters

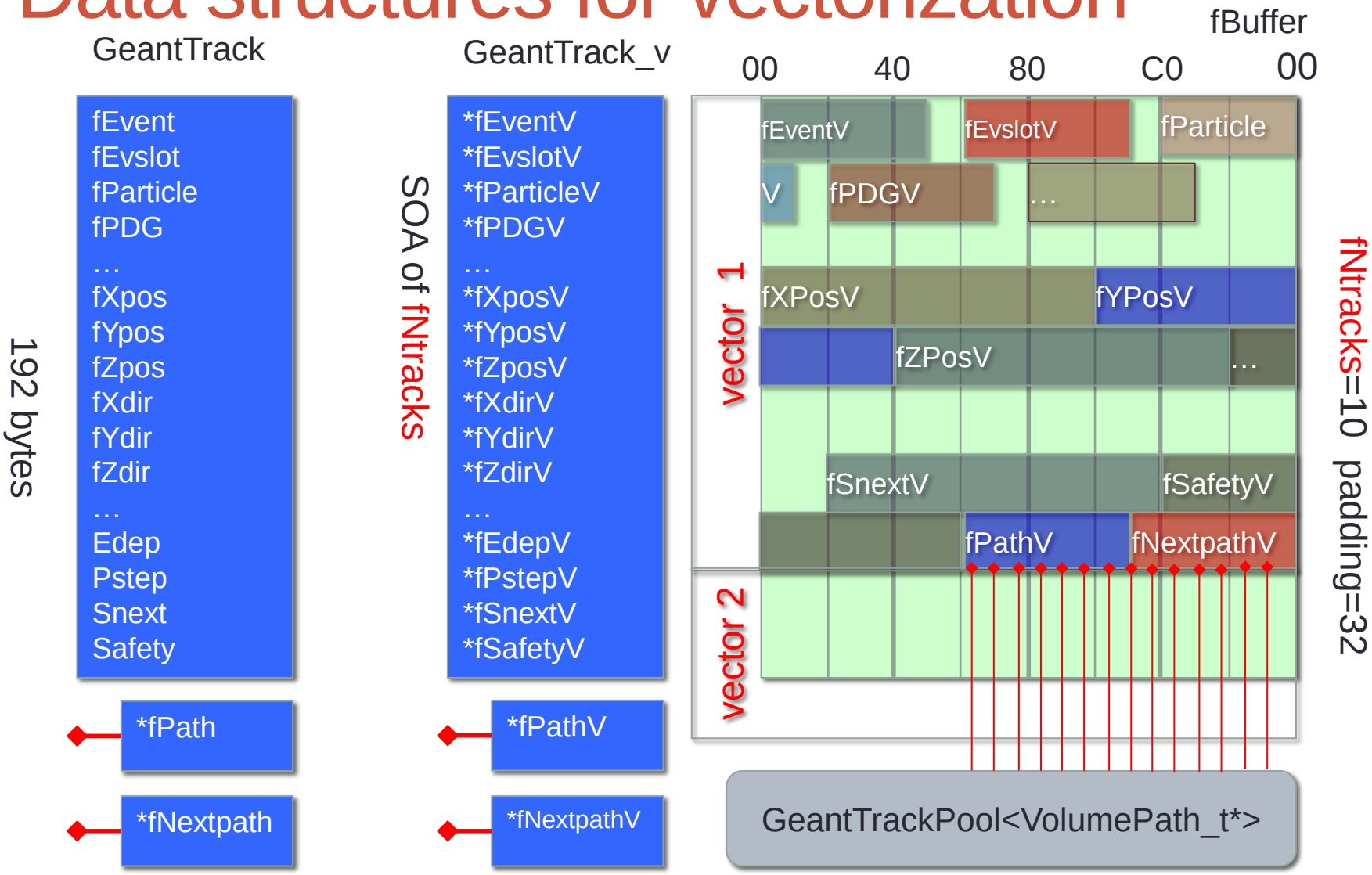
- Transforming existing G4 algorithms into “kernels”
- Support for vectorization
- Fast tabulated physics
- Template specializations
- Support for user fast simulation models

- Next generation geometry modeling
- Template specialized algorithms
- Re-usability of inlined “codelets”
- CPU/GPU transparent support
- Support for vectorization

Techniques for efficient algorithm vectorization and beyond: **see talk of Sandro Wenzel in this session**

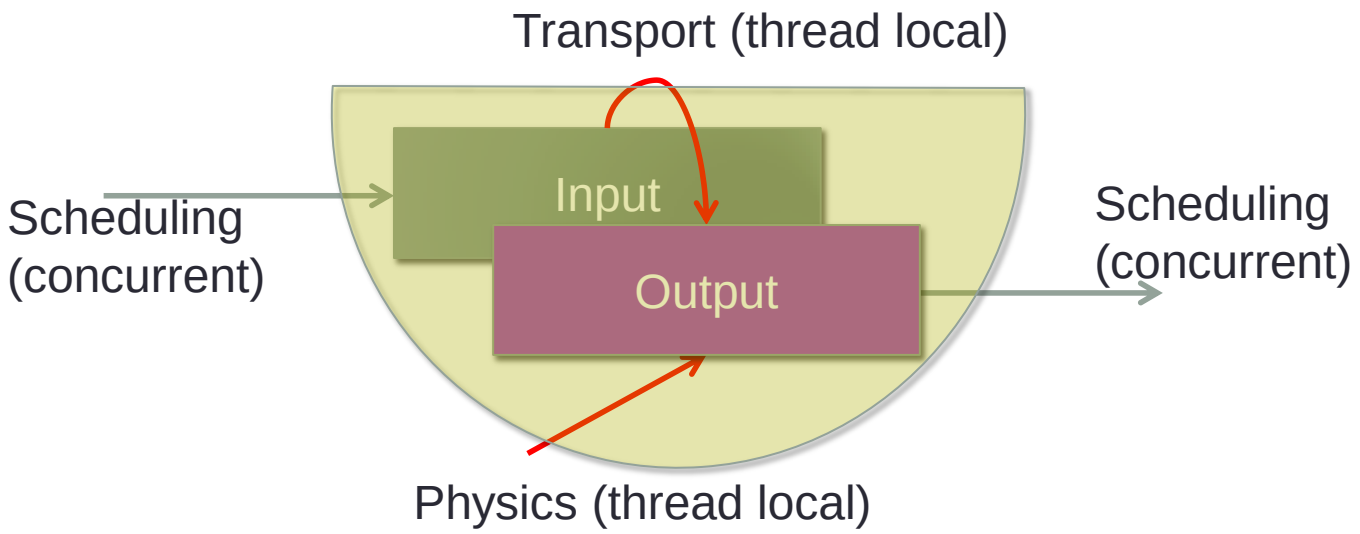
geant.web.cern.ch

Data structures for vectorization

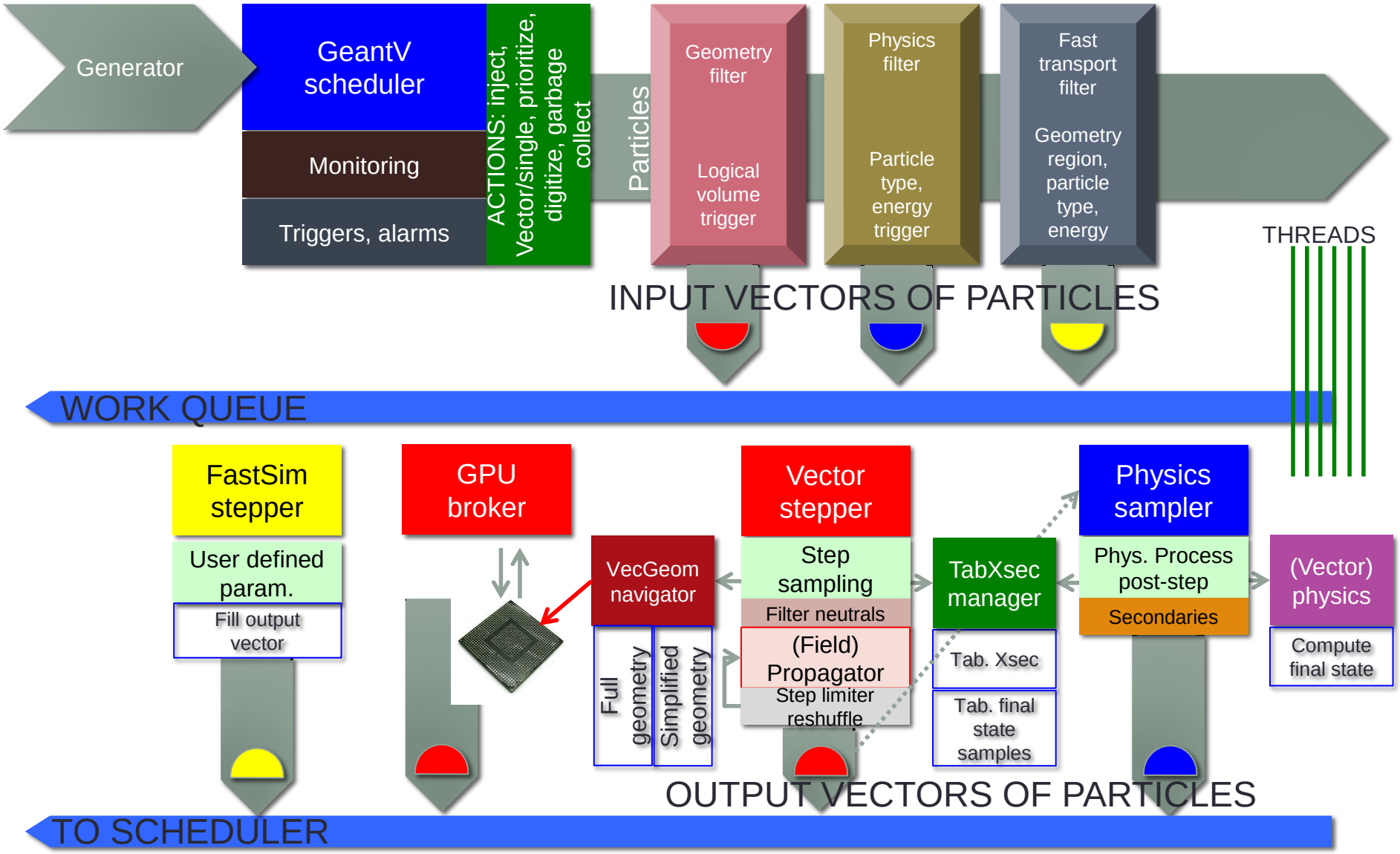


Track locality criteria

- Putting together tracks having some locality criteria -> baskets
  - Geometry: particles located in the same detector (logical) volume -> stepper
  - Physics: particles matching type/energy range -> physics processes
  - Custom: e.g. triggers for fast physics handover
- After some processing stage a particle is “stamped” for the next stage
  - Produced new particles added to the “basket” output
  - Currently stages are chained, in future the particle will be “released” for more efficient re-scheduling



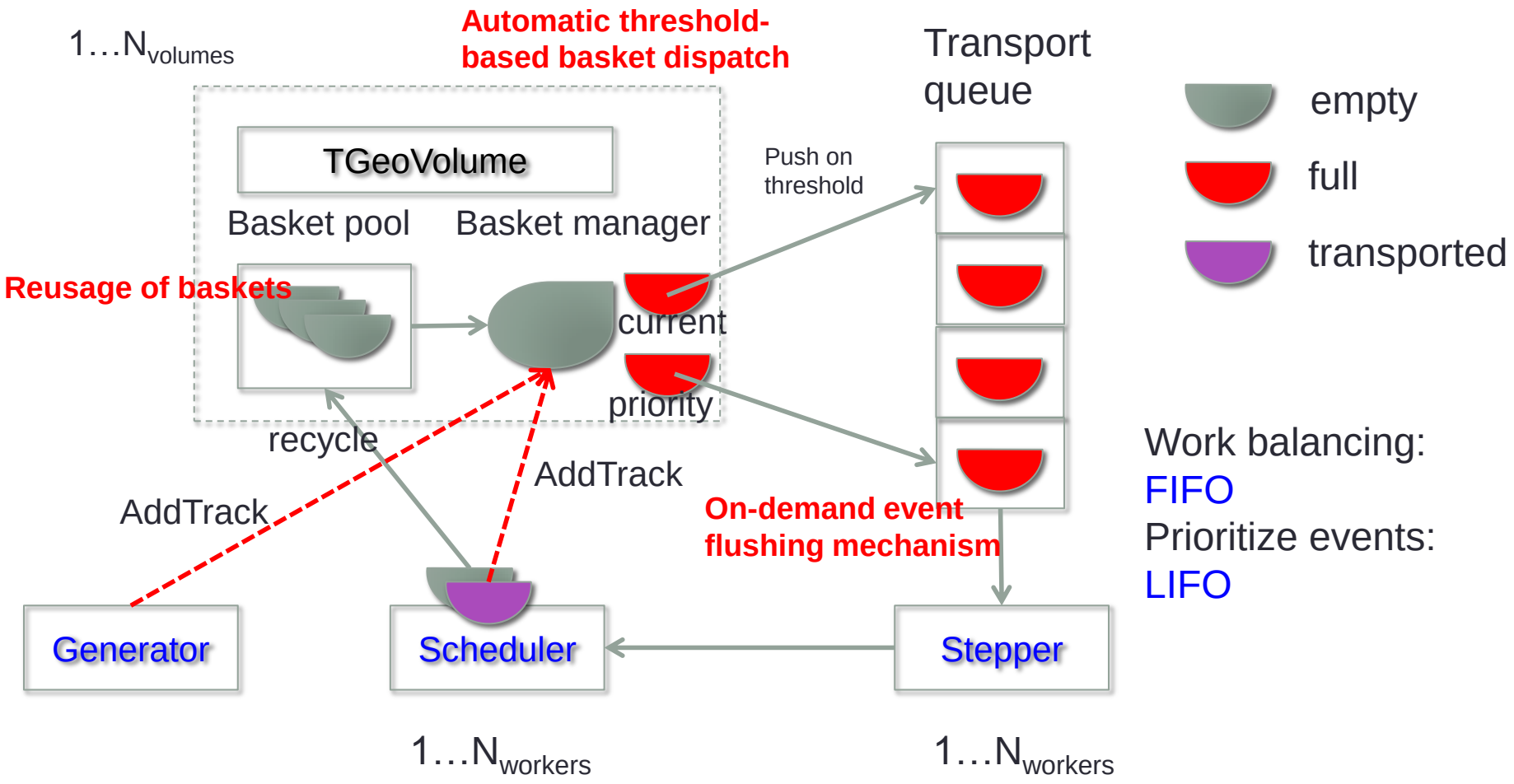
GeantV features



GPU Connector to the Vector Prototype

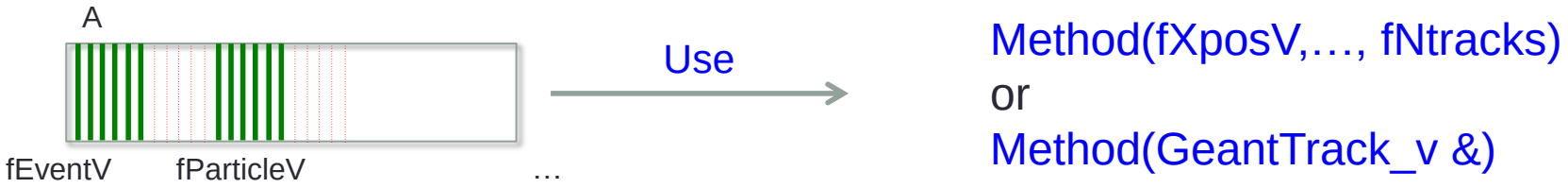
- Goals
  - Pick up baskets and select only tracks GPU can handle
  - Maximize kernel coherence
  - Adapt to GPU ‘ideal’ bucket size (very different from CPU)
    - Without hanging on to event too long
- Implementation
  - Stage particles in a set of buckets
    - Type(s) of bucket is customizable.
      - For example based on particle/energy that have a common (sub)set of physics models likely to apply
      - Keep order provided by main scheduler
  - Delay the start of a kernel/task until it has enough data or has not received any new data in a while
  - Start uploads after each basket processing to maximize overlap (even before the bucket is full) -> asynchronous transfers

Scheduling features

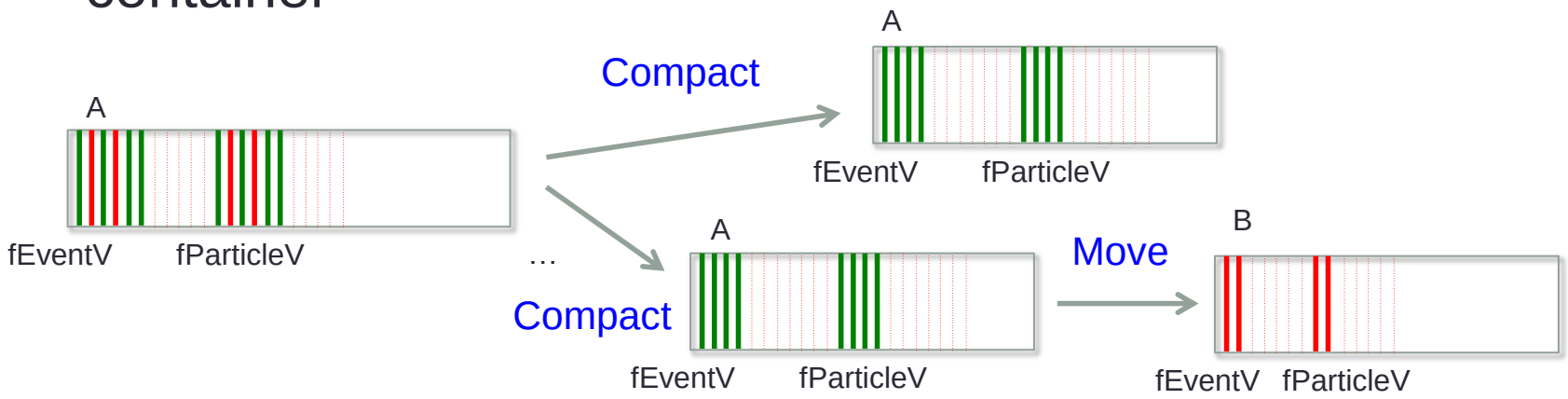


Challenges for vectorization

- Pre-requirement to use vectorized: contiguity and alignment of the track arrays



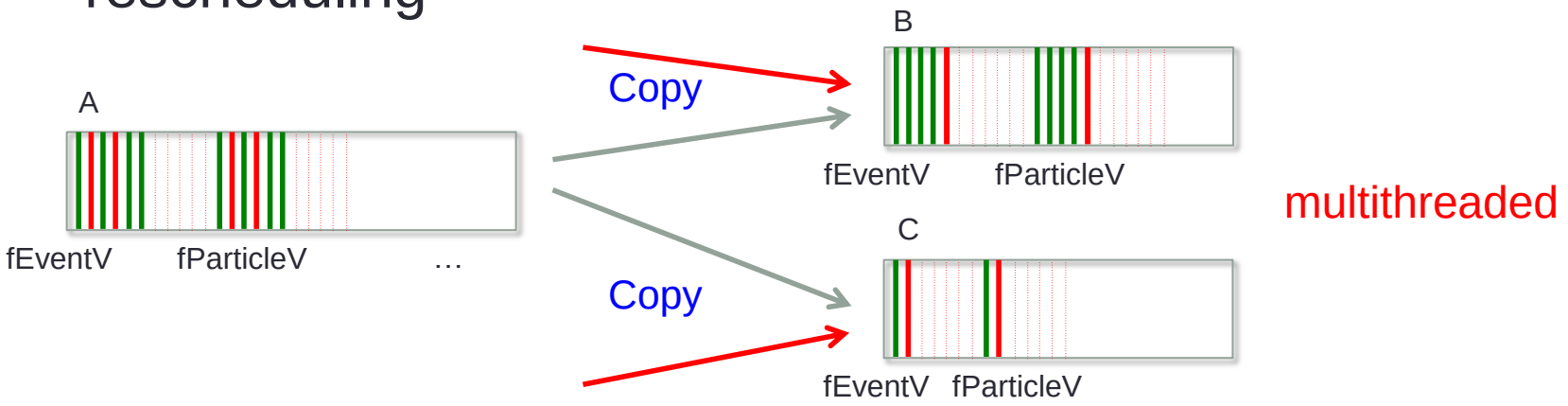
- During transport, tracks stop leaving holes in the container



Vector dispatching overheads

- Track selection according some criteria

- Tracks have to be copied to a receiver during rescheduling



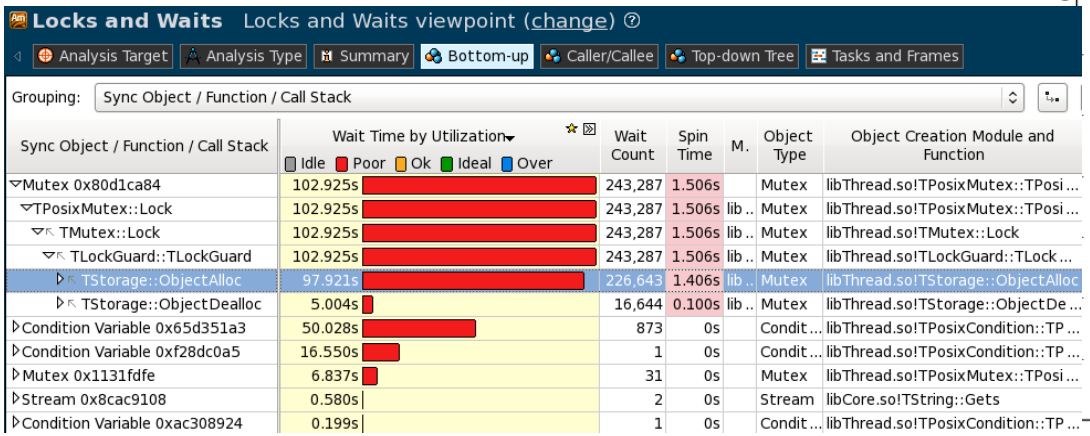
Scheduling actions

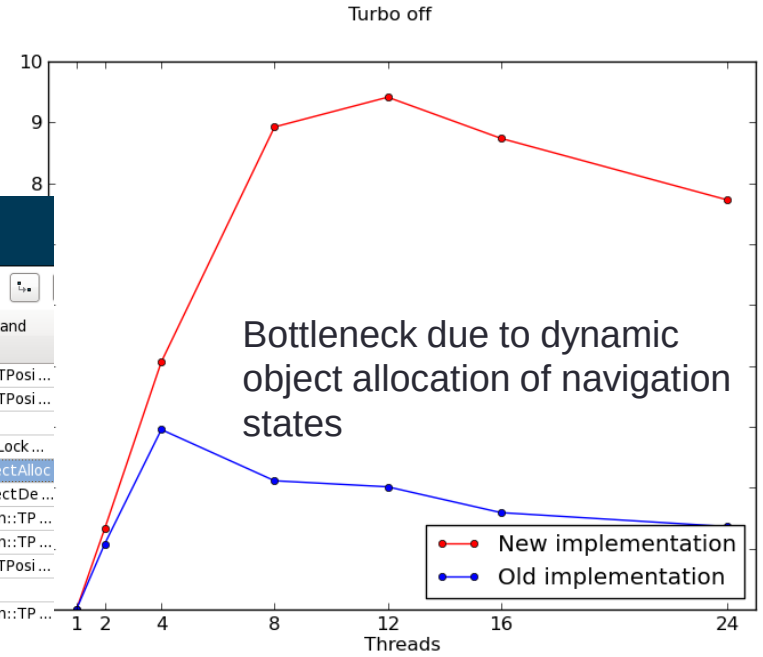
- The scheduler has to apply policies to:
  - Provide work balancing (concurrency)
    - Single work queue
  - Keep memory under control
    - Buffering limited number of events
    - Prioritizing events, prioritizing I/O
    - Issuing event flushing actions
  - Keep the vectors up (most of the time)
    - Optimize vector size
      - Too large: to many pending baskets
      - Too small: inefficient vectorization
    - Trigger postponing tracks or tracking with scalar algorithms
- Sensors/triggers
  - Work queue size thresholds
  - Memory threshold
  - Vector size threshold

Scalability for MT is challenging

- Performance is constantly monitored
  - Jenkins module run daily
- Allows detecting and fixing bottlenecks
- Amdahl still high due to criticity of basket-to-queue dispatching operations

1000 events with 100 tracks each, measured on a 24-core dual socket E5-2695 v2 @ 2.40GHz (IVB).

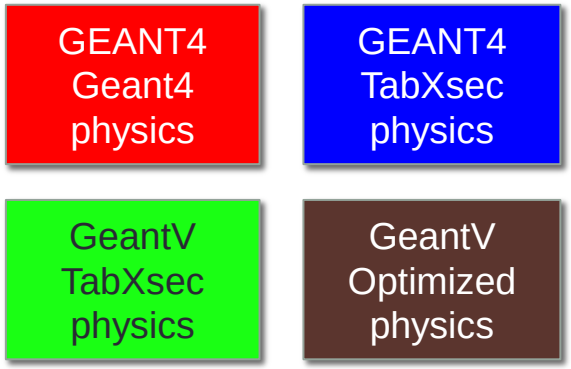




“Fast” physics and upgrades

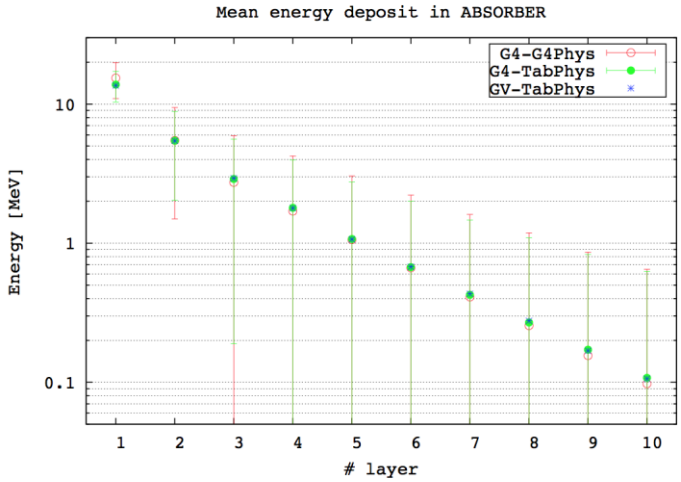
- Optimizing the performance of GEANT4 physics will have a long path
- **Goal:** compact, simple and realistic physics to study the prototype concepts and behavior
- **Requirements:** reproduce physics well enough to study the prototype behavior
  - energy deposit, track length, # steps, # secondary tracks, etc.
- **Implementation:**
  - tabulated vales of x-sections(+dE/dx) from any GEANT4 physics list for all particles and all elements over a flexible energy grid (EM and hadronic processes)
  - flexible number of final states for all particles, all active reactions, all elements are also extracted from GEANT4 and stored in tables

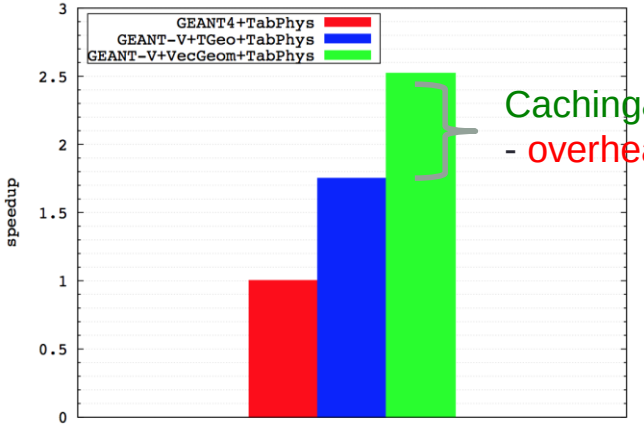
- **Status:**
  - a complete particle transport has been implemented based on these tables both behind the prototype and behind GEANT4
  - possible to test new concepts, performance relative to GEANT4 tracking
  - individual physics processes can be replaced by their optimized version when ready



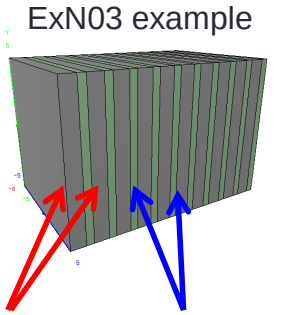
Preliminary performance checks

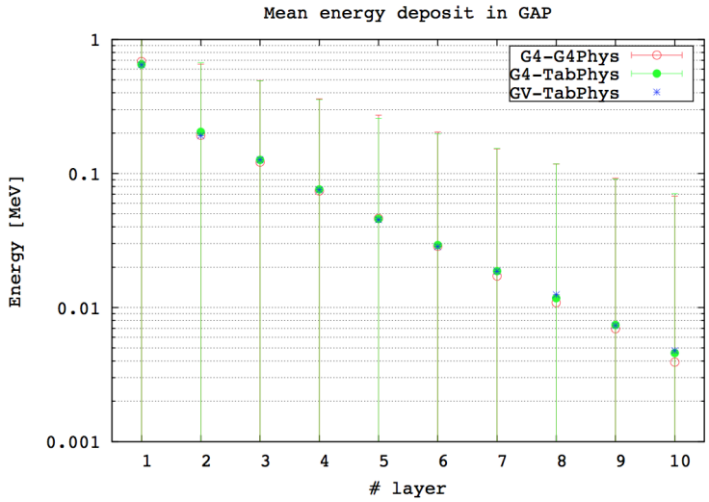
- Is overhead larger than gain?
- Simple example imported from GEANT4 novice examples
  - Scintillator+absorber calorimeter
  - 30 MeV to 30 GeV electrons, 100K primaries
  - Physics reproduced, small differences to be investigated for the highest energy
- Single thread performance expected to increase with the setup complexity
  - Evolve the example





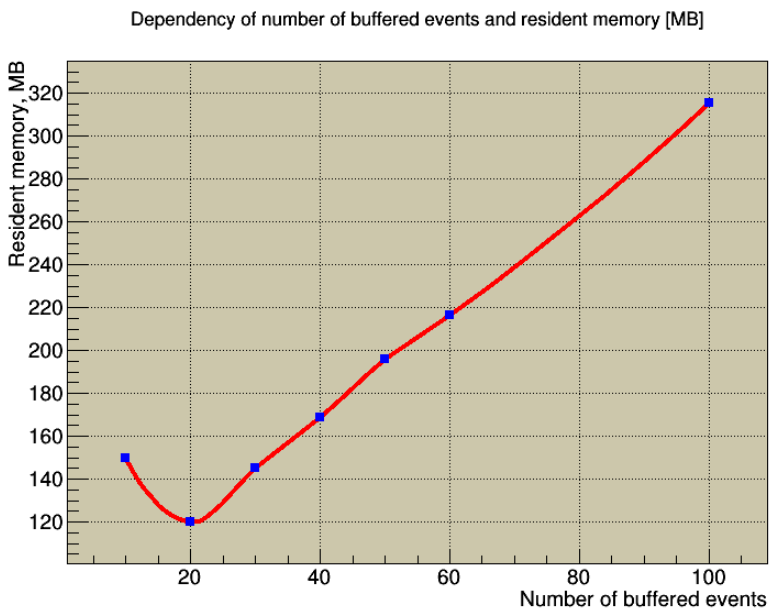
Caching&SIMD
- overheads

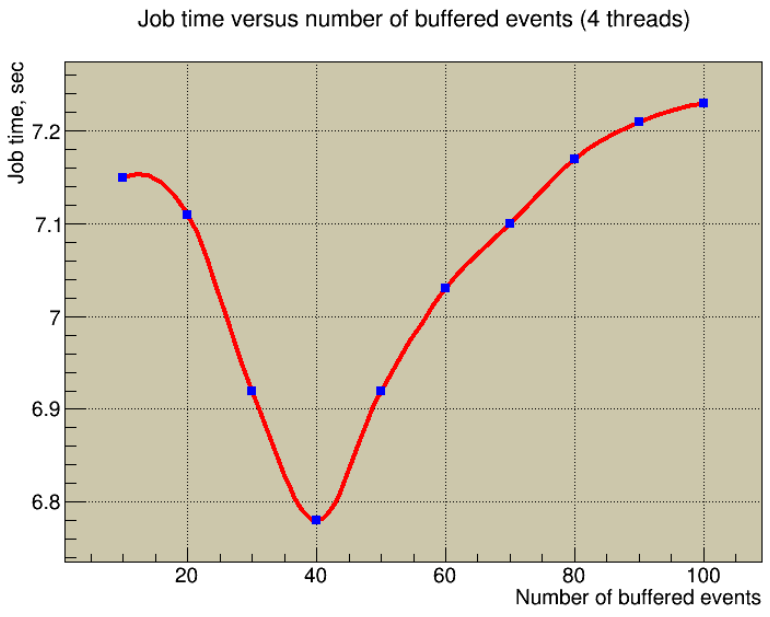




Parameters: memory buffer

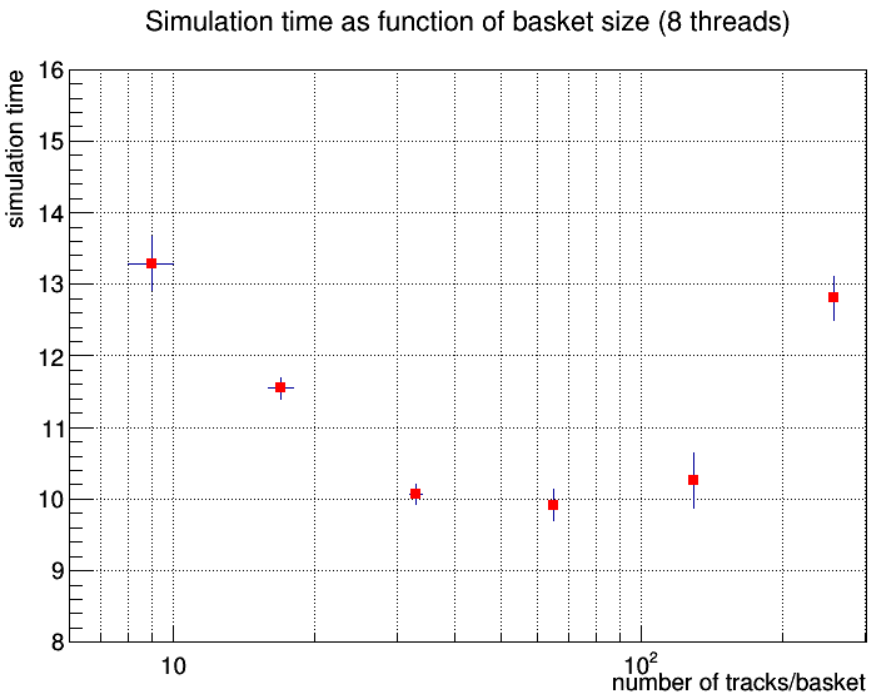
- Policy: keep fixed number of events in memory
  - Inject N_{buff} events at startup (from N_{total} to be simulated)
  - As an event gets flushed, inject a new one
- Maximum memory has an optimum at N_{buff}/N_{tot}~ 20%
- CPU performance is reached at N_{buff}/N_{tot}~ 40%
  - Gain for optimum value can reach 10%





Parameters: basket size

- The vector size is a major parameter of the model
  - Impacts on vectorization potential
- The optimum value depends on many parameters
  - Such as geometry complexity, physics
  - To be explored for several setups
- Small vectors = inefficient vectorization, dispatching becomes an overhead
- Large vectors = larger overheads for scatter/gather, more garbage collections (less transportable baskets)
- The differences in total simulation time can be as high as 30-40%
  - Aiming for an automatic adjustment of vector size per volume
  - Performing at least as good as the optimum for fixed vector size



GeantV & genetic algorithms

- Optimize GeantV scheduler model
  - Use genetic algorithms to find the optimum in the parameter space
  - Repeat for many setups with different geometry and physics configurations
  - Understand the patterns of the model and derive adaptative behavior for parameters (short learning phase followed by parameterized behavior)
- Model chromosomes: thresholds for prioritizing events, basket size, number of threads, threshold for switching to single track mode, size of event buffer
- Fitness function: minimize simulation time while keeping in predefined memory limits
- Currently investigating single parameter space
  - Understand the range to be scanned, expected behavior
  - Used in order to define the crossover, mutation, or other methods to evolve the genetic population

Summary

- Track scheduling is one of the core components of the GeantV project
  - Allowing to achieve performance from locality, fine grain parallelism and vectorization
- Vectorization has a cost
  - Overheads in vector reshuffling and gather/scatter have to be (much) smaller than the SIMD and locality gains
- The preliminary benchmarking shows important performance gains with respect to the classical transport approach
  - Simple geometry and physics so far
  - The gains expected to increase with the complexity
- The scheduling model is complex and its optimization difficult
  - Currently understanding the behavior of single parameters
  - Started to investigate an approach based on genetic algorithms