GeantV current status and plans

ANDREI GHEATA FOR THE GEANTV PROJECT

LHC DETECTOR SIMULATIONS: STATUS, NEEDS AND PROSPECTS

JUNE 27, 2017

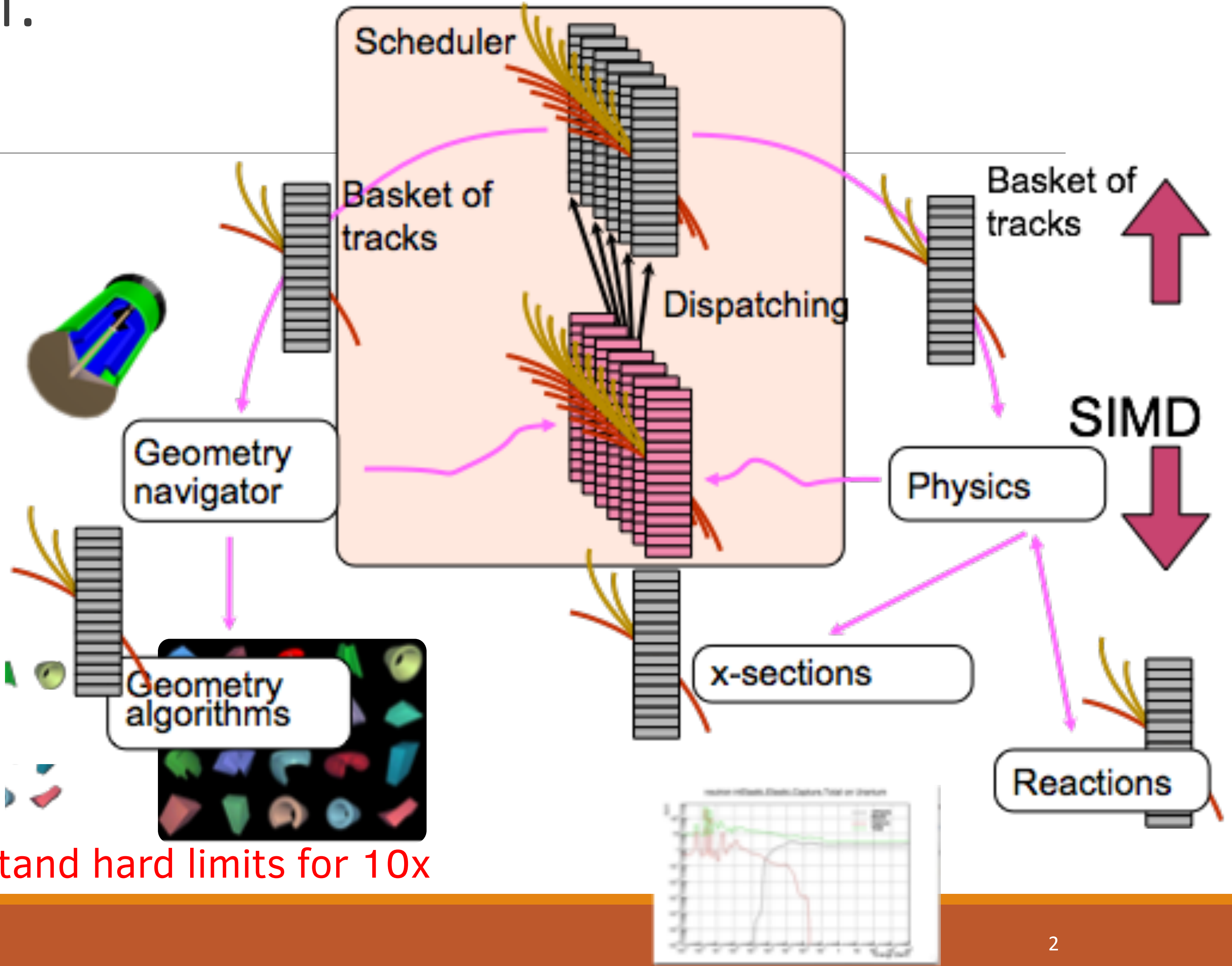
GeantV approach: boosting vectors

Transport particles in vectors ("baskets")

- Filter by geometry volume or physics process

Redesign library and workflow to target fine grain parallelism

Use an abstraction for vector types and their operations to achieve <u>portable</u> vectorization



Aim for a 3x-5x faster code, understand hard limits for 10x

Alpha release of GeantV (2017)

Providing stable interfaces and allowing experiments to "give it a try" with GeantV software

GeantV scheduler version 3

Finalized user interfaces

- Test case: experiment integration with parallel flow (CMSSW)

Vectorized Runge-Kutta propagator

Vectorized geometry

EM physics most/all processes for e+/e-/gamma in scalar mode

- first assessment on vectorization potential

Hadronic physics: Glauber-Gribov cross sections + low energy parameterisations, elastic scattering

Fast simulation hooks in GeantV, scope definition, integration and proof of concept based on examples

Full hit/MC truth cycle demonstrator

GPU demonstrator

Beta release of GeantV (2018)

Providing most of GeantV features/optimisations in terms of geometry, EM physics (partially hadronics), I/O and fast simulation. Allowing to actually integrate experimental simulations with GeantV as toolkit.

Production-quality scheduling, including error handling at the level of track/event, HPC demonstrator

Production-quality geometry supporting full features (construction and navigation) of Geant4 and ROOT

MC usage demonstrator based on realistic use cases. Integration with experiment SW.

EM physics – full shower physics (e+, e-, gamma), most CPU consuming models vectorized

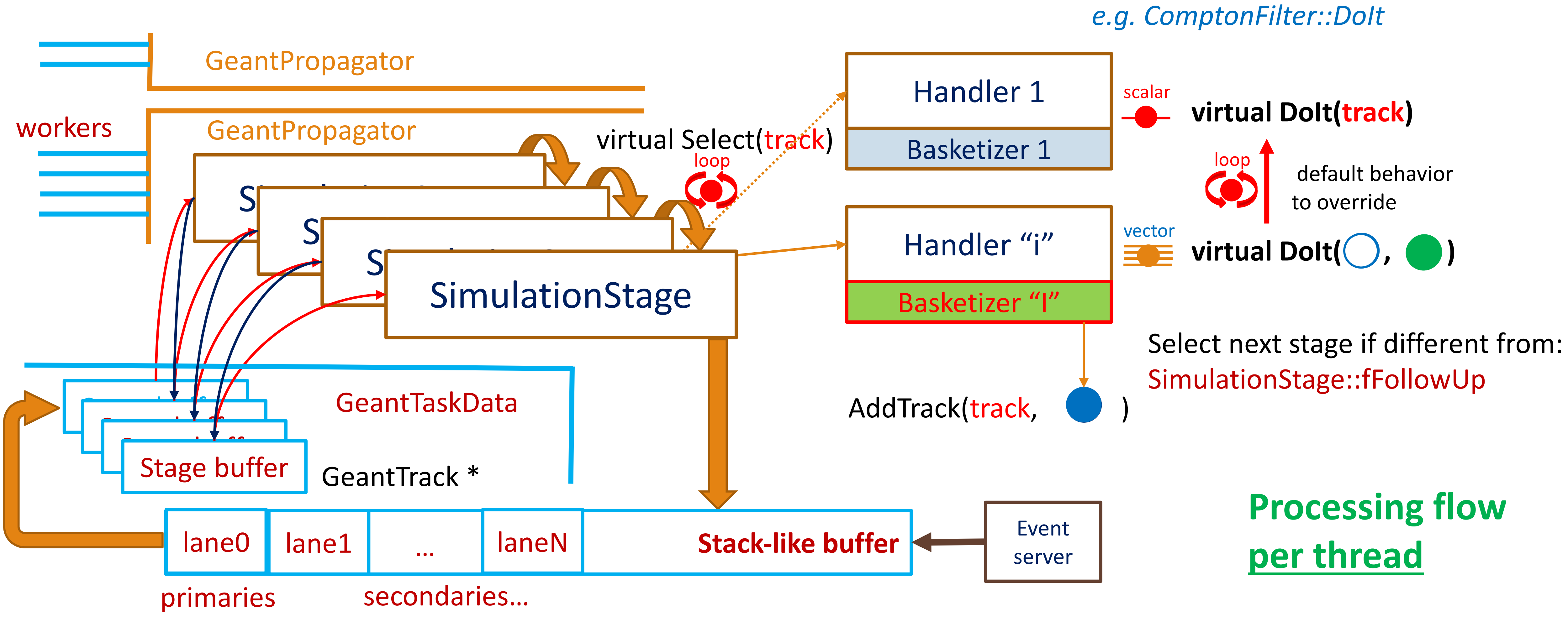
Hadronic physics: Bertini cascade, realistic model level and application level benchmarks

Integration of fast simulation with experimental frameworks, ML-based standalone tool + demonstrators for concrete cases

GeantV scheduler upgrade

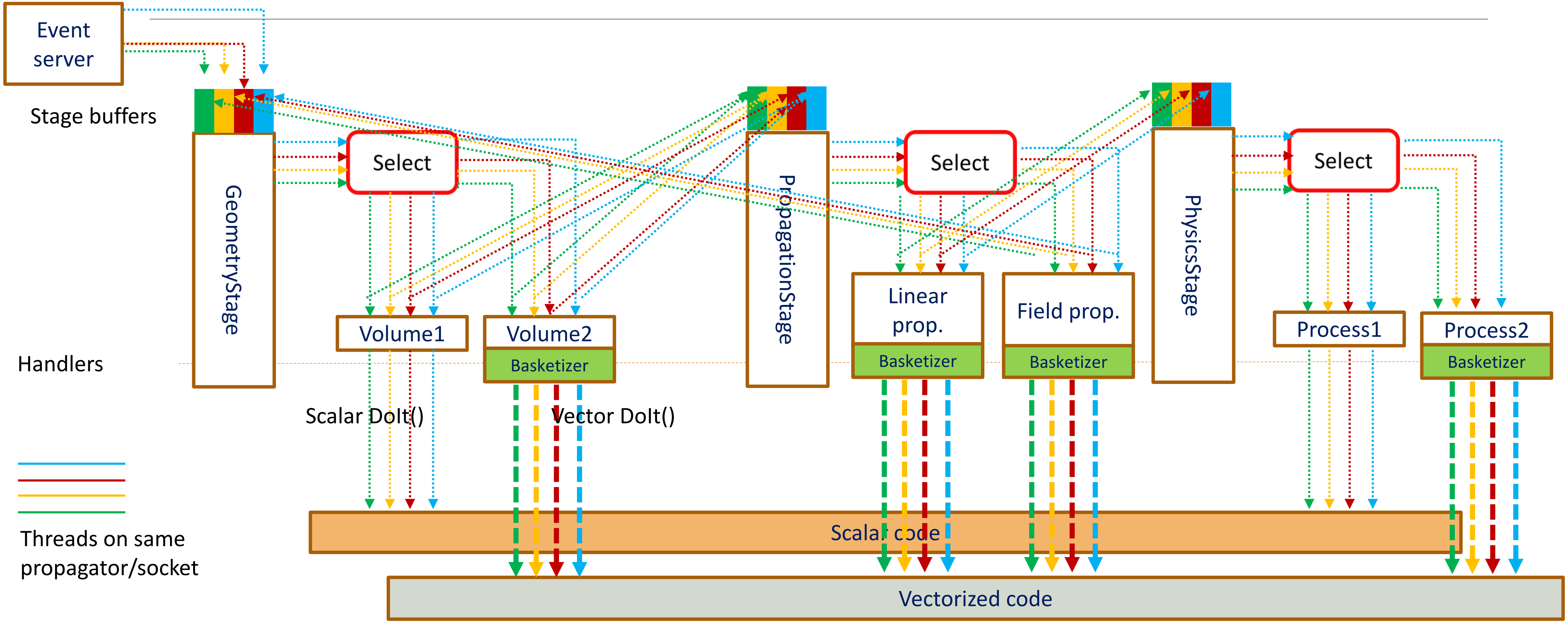
Scheduler version 2	Scheduler version 3
Geometry-centric basketizing approach	“Democratizing” the concept of basketizing to allow for physics multi-particle vectorization
SOA container handling: overheads for scatter/gather, reshuffling, concurrency	AOS handling in basketization, light SOA on demand for dispatching to vector code
“Avalanche” memory behavior: tracks are never released but only created, the full shower has to be kept in memory	More stack-like behavior, favoring transporting secondaries/low energy tracks with priority
Basket-driven concurrency based on non-local queues, adding contention points	Thread-local data and containers, relying less on common concurrency services (use my own data and containers as much as possible)
System-driven allocation of resources (threads, memory)	NUMA-aware allocation of resources, relying on topology discovery

GeantV version 3: A generic vector flow approach



Processing flow
per thread

Processing flow per propagator/NUMA node



Performance preliminary V3 vs. V2

V3 VERSUS V2, MEMORY, SCALABILITY, NUMA, TUNING KNOBS

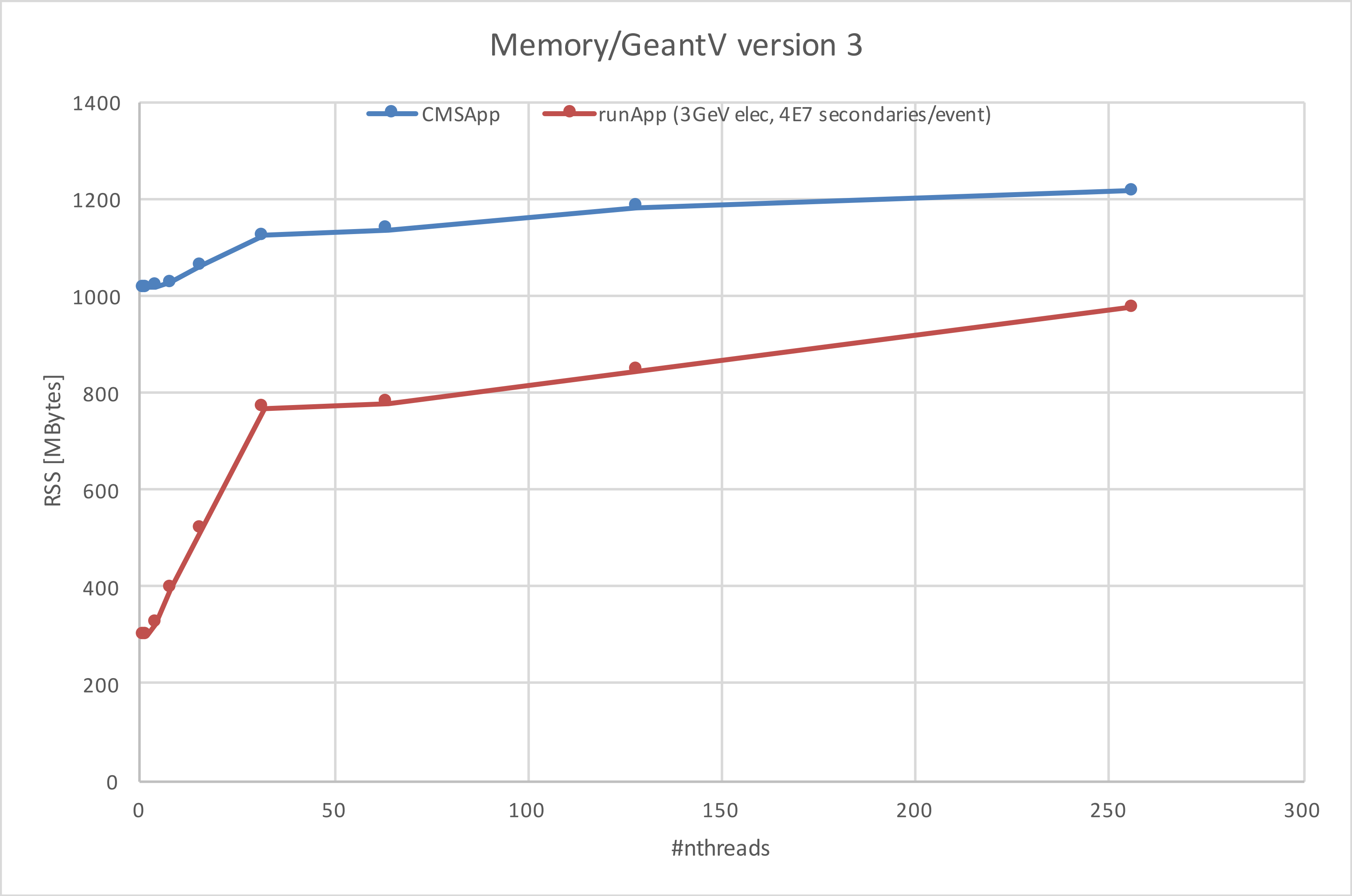
(FOR NOW JUST TABULATED PHYSICS, CMS SETUP & SIMPLE CALORIMETER)

Memory control

Stack-like control using a special buffer inserted in the stepping loop

- Higher generation secondaries flushed with priority

Very good behavior even for high number of threads/secondaries



NUMA awareness

Implemented using hwloc > 1.8

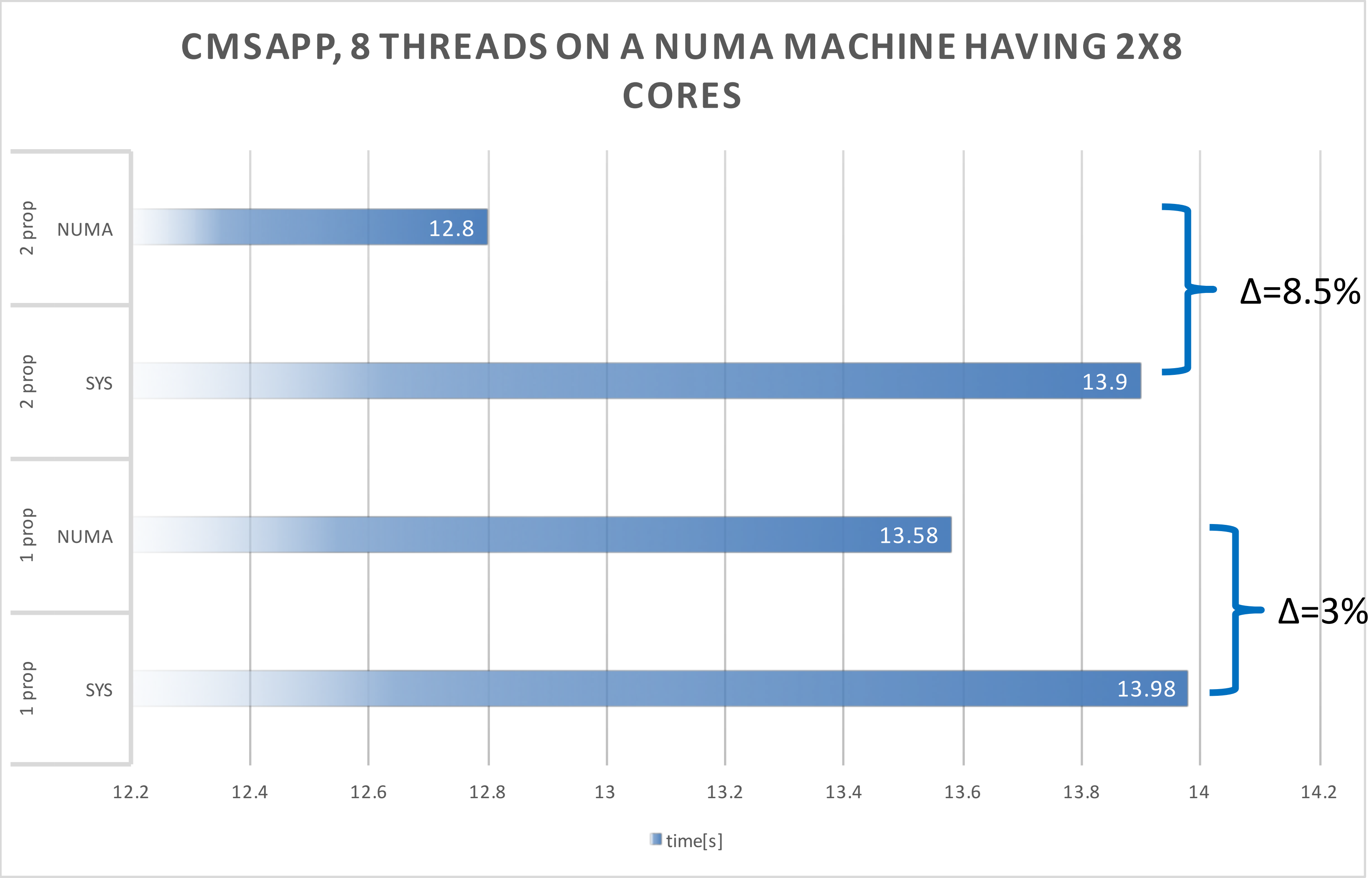
- Enumerating NUMA nodes, cores, CPU's
- Threads are bound to CPU's

A propagator will use threads bound to the same NUMA node

- More propagators can be bound to the same NUMA node

Compact policy used for threads on same propagator, scatter for distributing propagators on different nodes

Task data stage buffers, stack-like buffer, baskets and tracks bound to memory on the same node as the propagator owning the thread



Scalability

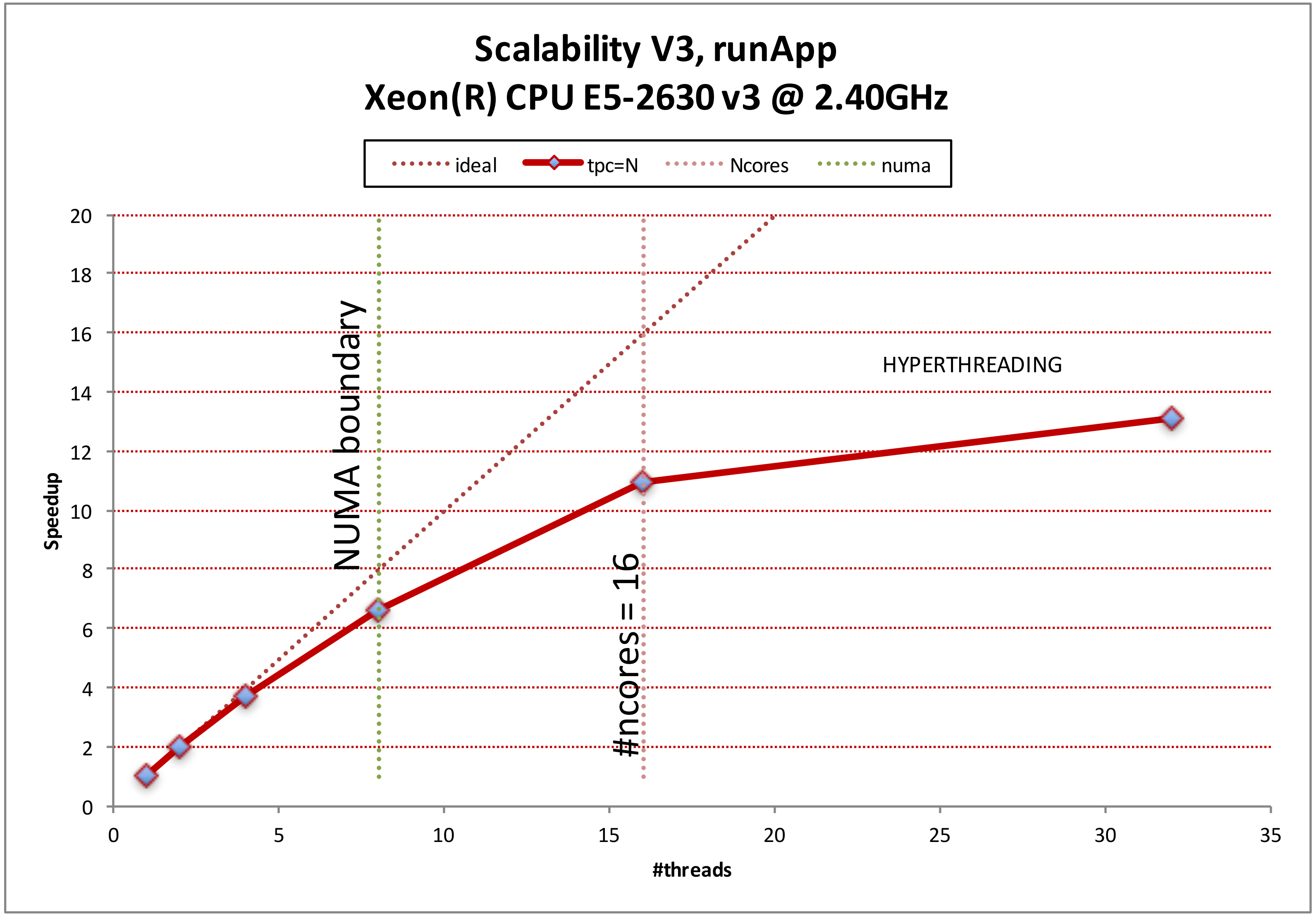
Not as good as expected

- Interaction between threads lesser, removed contingency points, SOA basketizing, no more basket queue

Profiling comparison N/2N threads does not reveal obvious hotspots

- To be further pursued

Memory operations are high in the profile, we expect picture to improve when having a more balanced scenario with more (vector) work on physics side.



Performance v3 versus v2

Relevant improvements in both single and multi-threaded mode

- Coming mostly from the increase of locality (simulation stages)
- Removal of SOA gather/scatter overheads
- NUMA awareness

Yardstick measurements to be redone

Physics goals

Full EM shower simulation in beta release

- Most important performance-related component for simulation
- Demonstrate important gains due to vectorization + locality (treatment of baskets)

Hadronics covering up to 10 GeV

- Glauber-Gribov cross sections (elastic,inelastic,total) + low energy neutron/pion parameterization (Barashenkov)
- Elastic scattering: rewrite from scratch, converging to a common version for GeantV and Geant4
- Bertini cascade extraction from Geant4
- R&D on evolution towards other promising models (e.g. EPOS)
- The EM component for hadronic showers (?)
- Fast simulation allowing functionality as in current Geant4, plus independent generic module based on ML with concrete examples

Status of EM shower simulation with V3

10^4 100 [GeV] e- in ATLAS bar. simpl. cal. : 50 layers of [2.3 mm Pb + 5.7 mm lAr]

	e^-/e^+: ionisation, bremsstrahlung; γ: Compton, conversion					
	GeantV		**Geant4**			
material	E_{dep}[GeV]	length [m]	E_{dep}[GeV]	rms [GeV]	length [m]	rms [m]
Pb	65.401	47.518	65.397	1.139	47.517	0.769
lAr	24.987	116.774	24.987	0.419	116.777	1.771

Mean number of :

gamma	38520	38515
electron	960734	960484
positron	5252	5253
charged steps	1069957	1069700
neutral steps	5176175	5175038

Coming soon: GS MSC integration, being now validated against Geant4

Plans for vectorizing physics

Goal: vectorizing the final state sampling methods of the physics models

Handlers will be automatically created for each different model

The post-step action stage will select tracks for a given model

The vector version of final state sampling method of the models will be possible to invoke

Performance assessment: switching on/off basketizing per model handler

User interfaces – GeantV impact on user framework

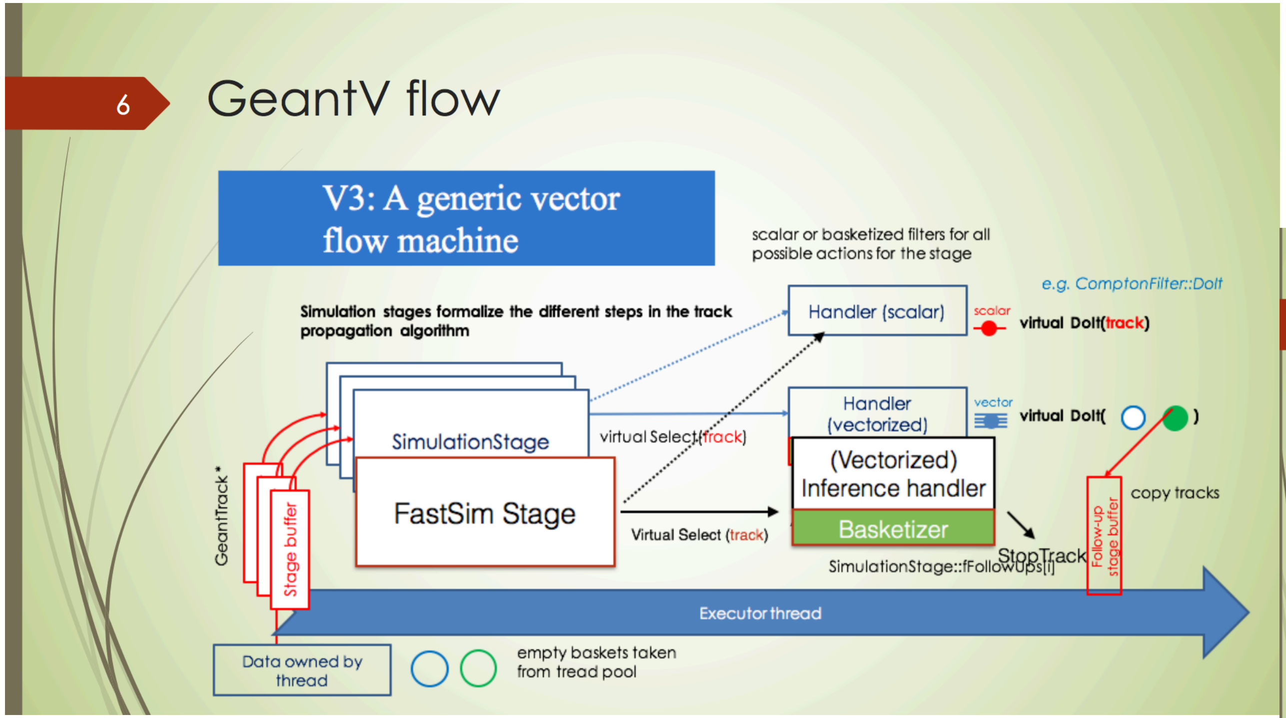
New features:

- Multi-threaded, multiple events in flight
- Concurrent scoring
- Multi-particle interfaces, tracks from multiple events mixed
- Possibility to vectorize time consuming user code (digitization)
- Concurrent digitization + merging of digits
- Multi-threaded handling of data structures
- Concurrent I/O

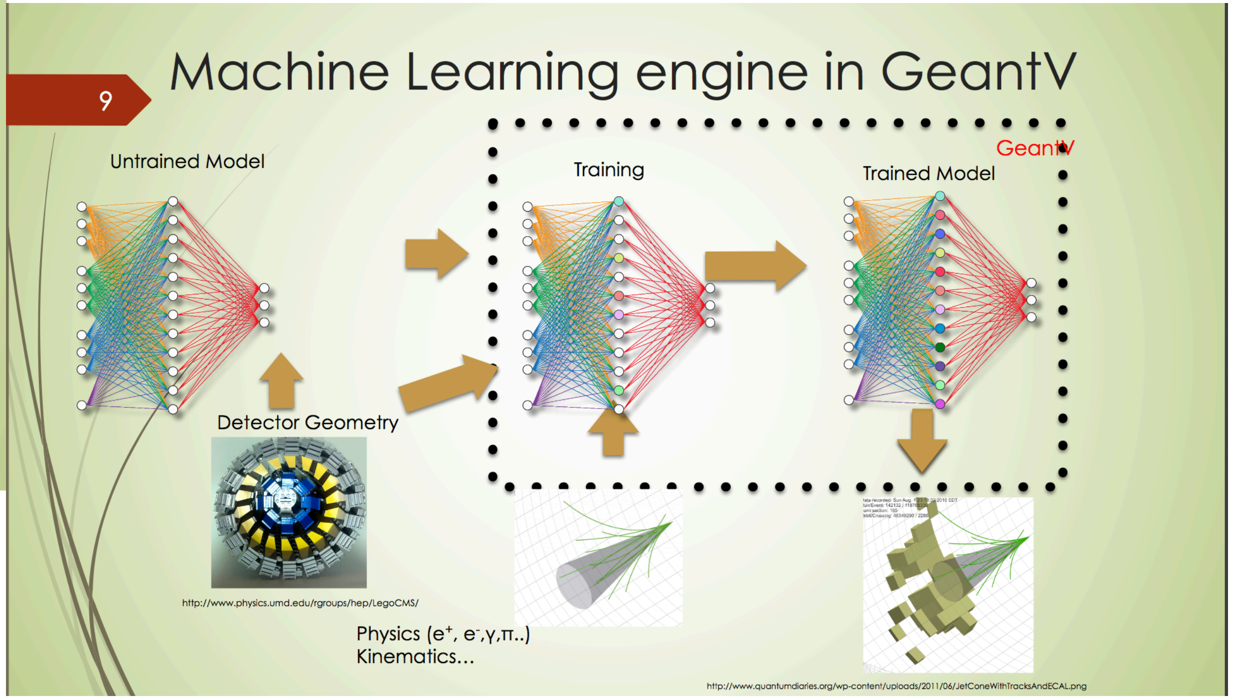
GeantV support:

- UI callbacks similar to Geant4, simplified access to state via track information
- Task data whiteboard providing hooks for user-defined data (no concurrent access on task data)
- Hits/digits concurrent factory, allowing to pre-alocate and use custom user data
- Concurrent I/O mechanism (now in ROOT)
- Vectorization API + backends (VecCore)
- Task-based parallelism to integrate with user task-based frameworks (e.g CMSSW)
- Event slot – based storage: fixed number of in-flight events, allowing to pre-allocate data on a limited number of slots
- User API for merging digits

Fast simulation



S. Vallecorsa



Next priority work (end of summer)

Integration with RP: full shower simulation with scheduler V3

- Most models already integrated with master ✓
- MSC with v3 ✓
- Photoelectric
- Integration/testing new RK propagator (scalar/vector)
- Performance tuning V3 + yardstick measurements GeantV vs. Geant4 extended to RP

Geometry vectorization activation for v3, fixes to fully match Geant4 raytracing in complex geometry

Finalize user interfaces

Hadronics

- Include Glauber-Gribov cross sections, finish initial development of elastic scattering process, evaluate feasibility of model “extraction” from Geant4 (e.g. Bertini)