FAST DETECTOR SIMULATION AND THE GEANT-V PROJECT

Andrei Gheata, CERN

ACAT 2014

Prague, 1-5 September 2014

Disclaimer

- The numbers presented here are extracted from several sources, not necessary the most up to date
- I focused just on LHC experiments for the fast simulation part, partially due to missing the time to extend my search

Outlook: (fast) simulation trends

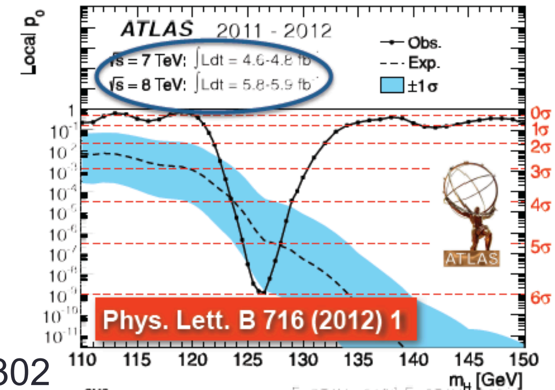
- Simulation@LHC startup
- Ramping up challenges
- Flavors of fast simulation
- Fast simulation in experiments
- Frameworks, integration, trends
- Summary I

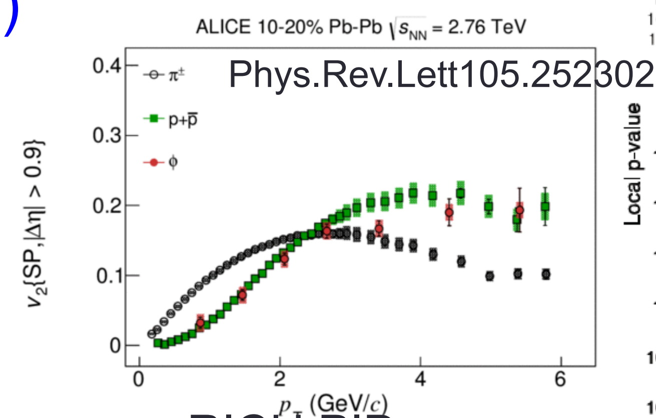
Outlook: GeantV challenges

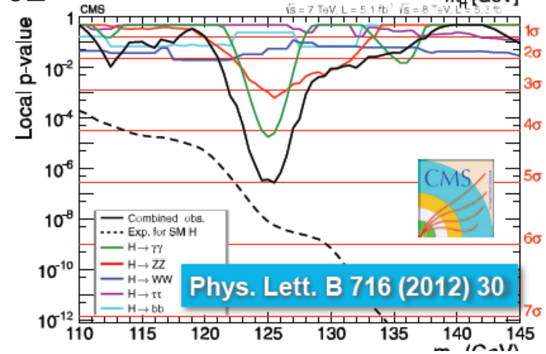
- The project
- Particle scheduling concept & challenges
- Optimizing geometry and physics computation
- User access to low level optimizations
- Crossbreeding slow and fast
- Roadmap and challenges
- Summary II

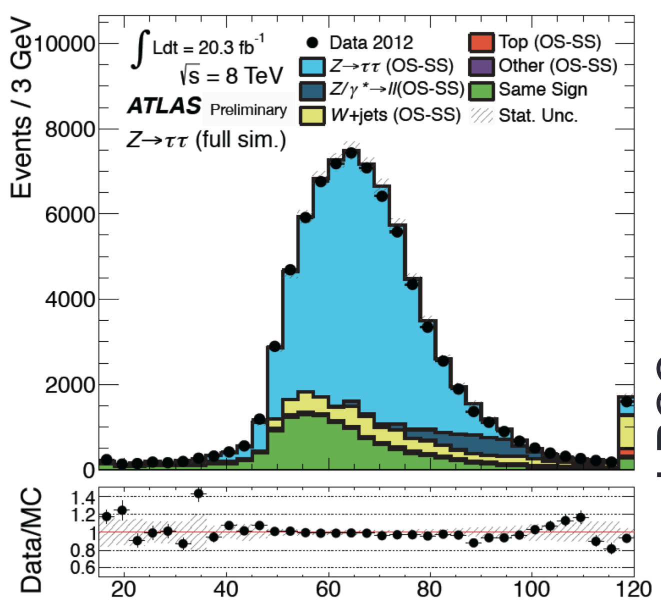
Simulation@LHC startup

- Excellent validation for the simulation frameworks on most observables
- State of the art GEANT4 physics
- A big (but challenging) success



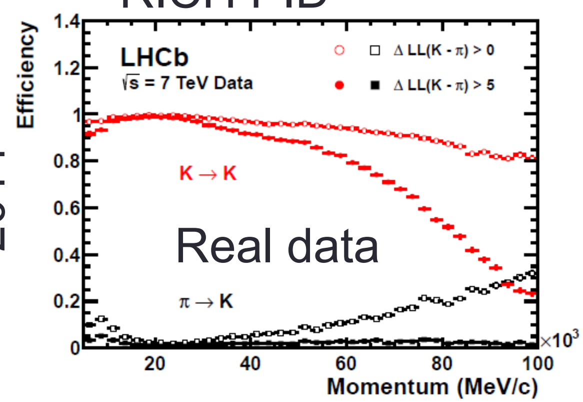


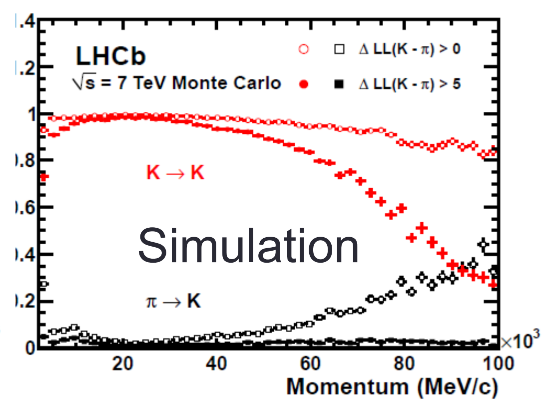




LPCC 2014

RICH PID

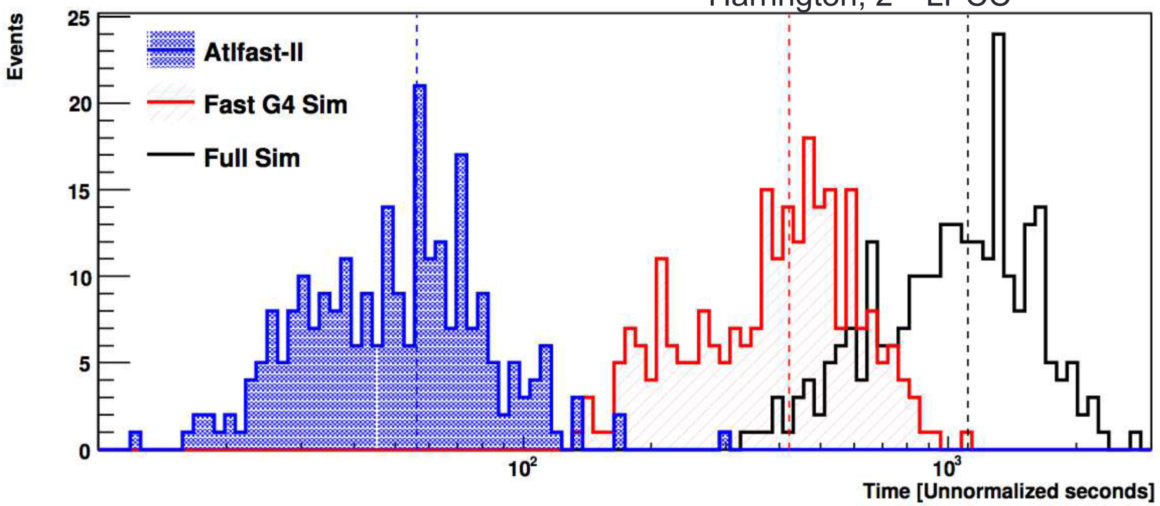




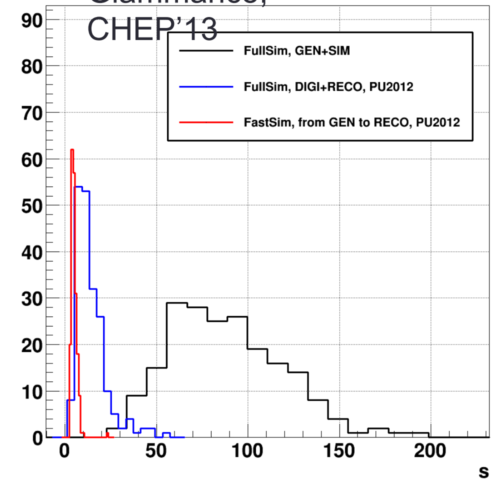
The need for simulated samples

- Simulation indispensable in the experiment's design phase
  - Simplified setup, good physics modeling
  - Detector R&D – detailed simulation
- Large samples are generally needed to reduce systematic errors
- In the study of rare signals, large-statistics samples are needed
  - Detailed simulation becomes prohibitive to get significant signal
    - Simulate signal events and merge with sampled background
    - Use background parameterizations
- Time and resources becoming limiting factors

Harrington, 2nd LPCC



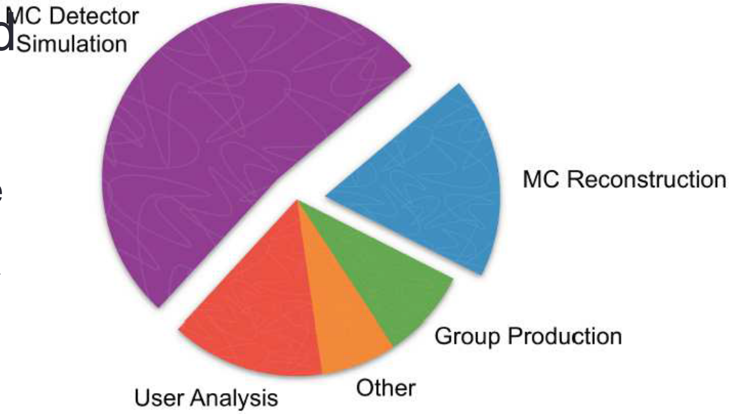
Giammanco, CHEP'13



Run1 simulation - a CPU challenge

- ATLAS: several billion events/year (~1/2 grid resources)
  - Aiming for 1/1 ratio (1/3 full + 2/3 fast)
    - Last MC production (~7 Billion events) managed the opposite
  - Up to 6 minutes/event MB, largely dominated by calorimetry
- CMS: several billion events/year
  - ~20-100 sec/event full, ~1 sec/event fast
- LHCb: few billion events produced
  - 100/1 (rare signals) 1/100 (rest)
  - Simulation time: 1 min-1 hour/event range
  - Digitization: less than 1% of transport
- ALICE: ~1 Billion simulated events (full)
  - Taking more than 50% of GRID resources
  - p-p at ~60s/event, Pb-Pb MB at ~ 10 min/event
    - Transport and generation: 70% (mostly ZDC)
    - Digitization: 30% (mostly TPC – ExB, diffusion)

ATLAS GRID CPU utilization



LHCb GRID usage	2013	2016
Sim	64.5%	63%
User	20.2%	8%
Rest (str, repro, rec)	15.3%	29%

ALICE MC events per year

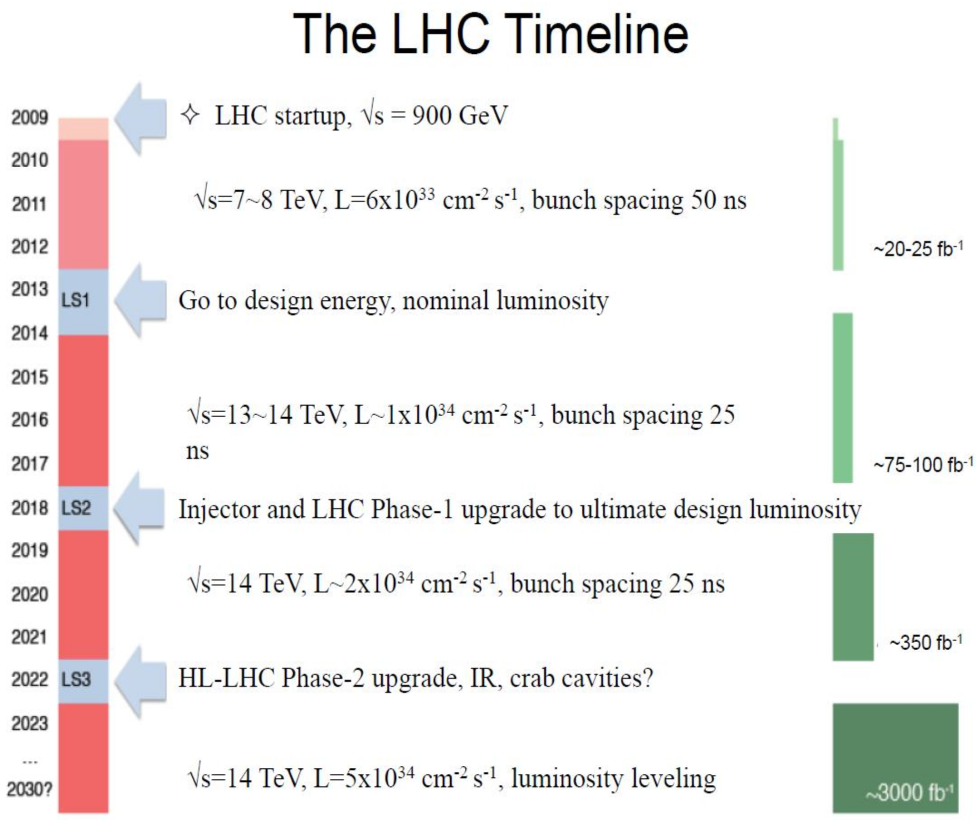
	2010	2011	2012	2013
pp	876 M	331 M	589 M	557 M
p-Pb				340 M
Pb-Pb	1.1 M	26.4 M	44 M	74 M

The LHC "pressure"

- Number of collisions increasing
  - 3-5x for Run 2
  - 10x for Run 3
  - 100x for HL-LHC
- Increase in energy and pileup
  - Non-negligible impact in simulation time

Hildreth, Ivanchenko – CHEP13

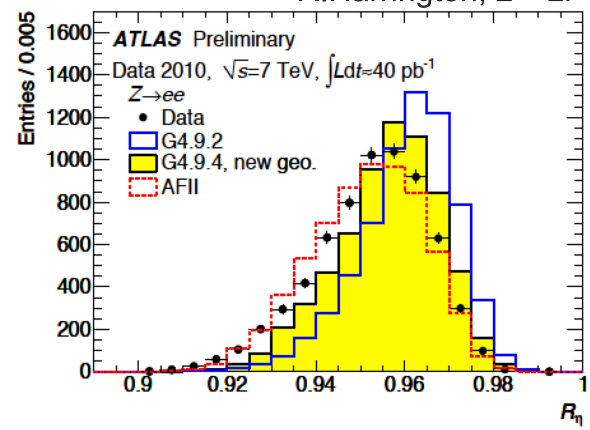
Process	8 TeV	14 TeV
MinBias	19.3s	21.5s 111%
Z→e+e-	50.9s	116.9s 230%
ttbar	87.1s	115.8s 133%



- Assuming the same ratio simulated/data, we have a potential problem…

Fast Simulation – taking the shortcut

R.Harrington, 2nd LPCC

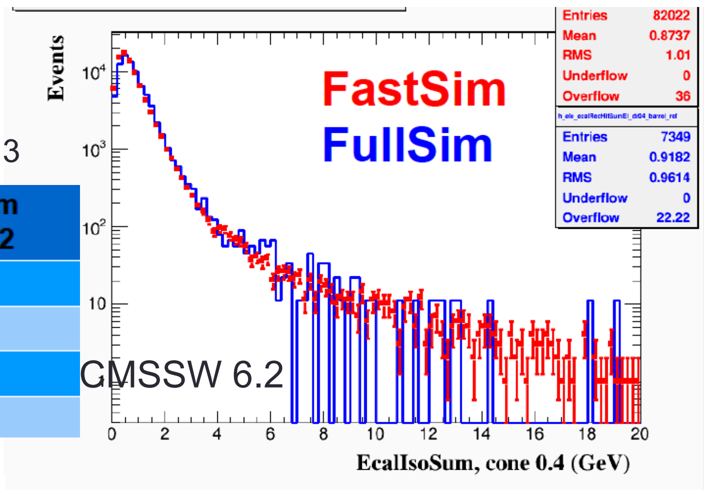


Energy ratio R_η in a $\Delta\eta \times \Delta\phi = 3 \times 7$ cells cluster with respect to a 7×7 cells cluster size in the bulk EM calorimeter layer 2

- Compensate the lack of time and resources to produce MC samples by a faster approach
  - Increase in throughput of O(10-100)
- Fast simulation is an option for many analyses
- Price: physics performance, to be considered case by case

A.Giammanco, CHEP 2013

ttbar @ 13 TeV	FullSim no PU	FastSim no PU	FullSim PU2012	FastSim PU2012
Generator (Pythia)	0.02	same	same	same
Detector simulation	88	0.20	same as no PU	0.88
Digitization	0.7	0.24	3.2	0.30
Reconstruction	1.9	1.2	10.6	2.8



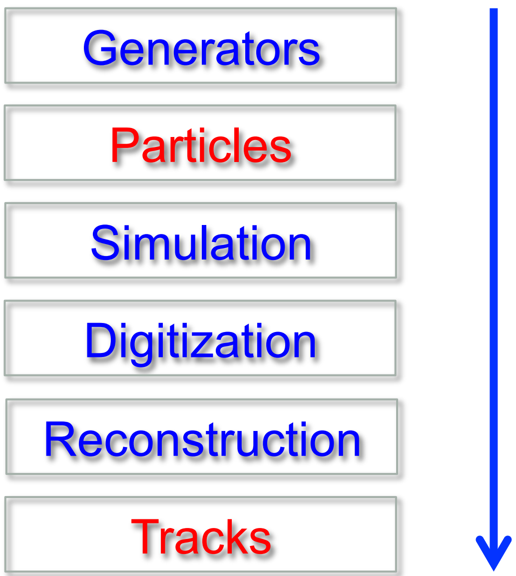
A plethora of FastSim approaches

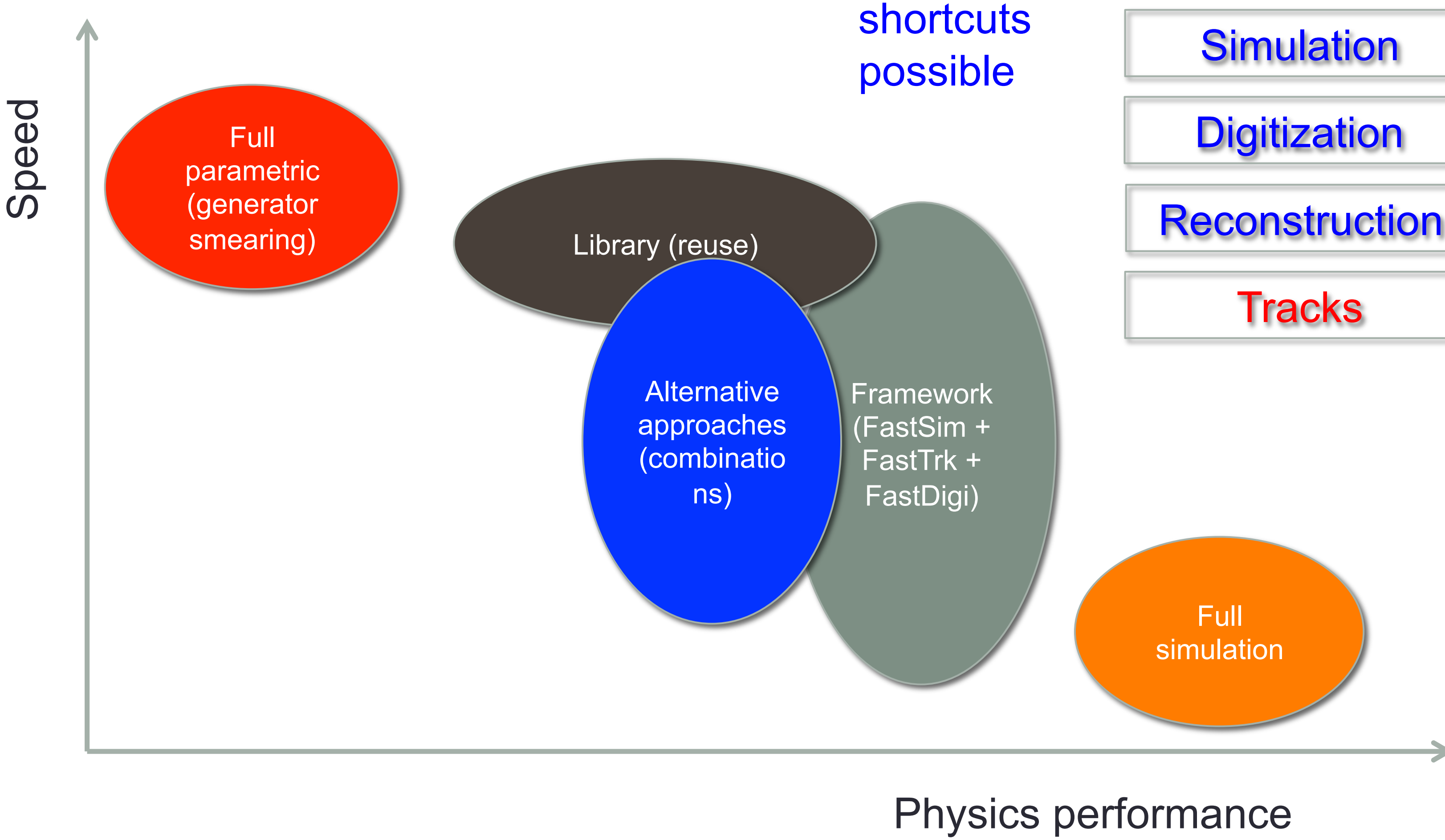
Replacing totally or partially detailed simulation components with parameterizations or pre-generated samples.

- Parameterizations smearing from generator level
  - Such as PGS or DELPHES – generic, or ATLFAST-I, CMSJET, ALICE standalone parametric simulation - specific
- Re-using events (full sim or data) – library approach
  - Embedding signal into background simulations, or merging simulation on data background (e.g in ALICE simulation framework)
  - Replacing costly physics objects with pre-simulated ones (e.g. Frozen Showers in ATLAS)
- Alternative approaches – any combinations of the following
  - Selective parameterizations (material/interaction), filtering on different criteria
  - Fast reconstruction geometry replacing full geometry selectively
  - Fast tracking
  - Combining simulations at different stages (slow+fast)
  - CMS FastSim, ATLFAST2
- Framework approaches
  - Sequence of fast/slow simulation with fast/slow tracking + fast/slow digitization
  - Combining GEANT & fast modules in a single session
    - “Slow” simulation toolkit as a component

Flavors overview

Any shortcuts possible



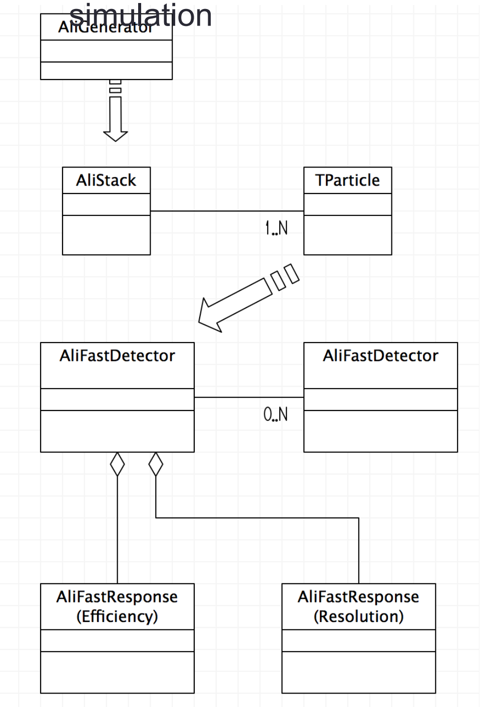


Full parametric FastSim

Used in general for TDR phase, good in describing bulk effects of the detector on reconstructed particle observables, very fast, suitable for generating large statistics background samples, not very used in analysis

- PGS, Delphes
  - Parameterization of detector efficiencies and resolution
  - Generic detector: tracking system (mag. field), calorimeters, muon system
- ATLFAST-I
  - Parameterizing kinematics w/o most detector effects
  - Smearing based on measured detector resolutions (generally Pt, eta, for particles and jets)
  - Early need for better describing realistic reco. efficiencies, specializing per track
- CMSJET
  - FORTRAN code parameterizing on jet observables
  - Not used anymore since ~2005. Too rough even for the TDR
- ALICE full parametric simulation
  - Early physics performance studies

ALICE full parametric simulation

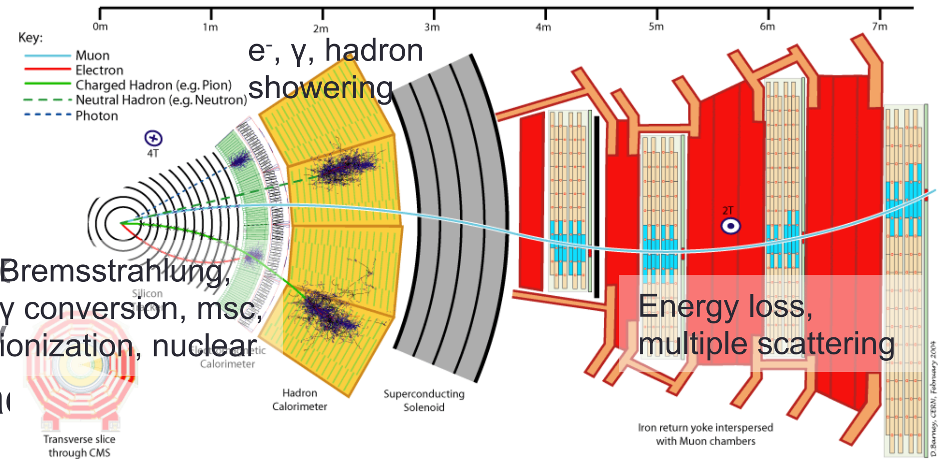


Combined parametric detector simulation

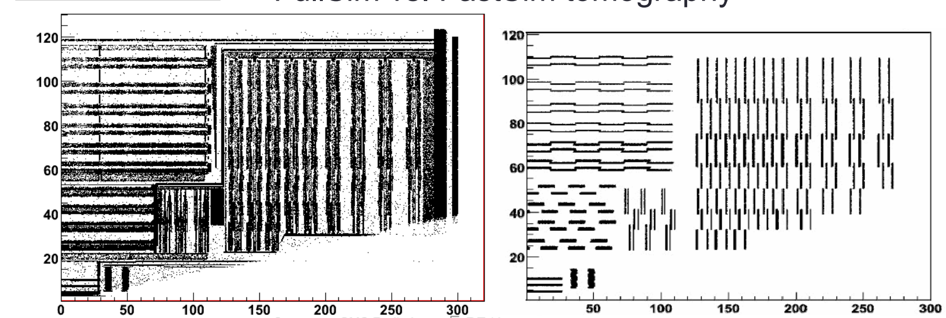
- Low level parameterization of material effects per track
  - No “microscopic” track propagation
  - No full parameterization smear at generator level
- Generation of low level tracking objects (hits, digits, clusters
  - Applying detector response via “digitizers”
    - In the same way as for full simulation
- Combining optionally with fast tracking modules
  - Replacing normal seeding with MC truth
- Examples: CMS FastSim, Atlfast2(F)

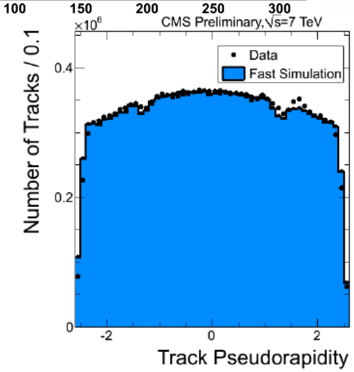
CMS FastSim

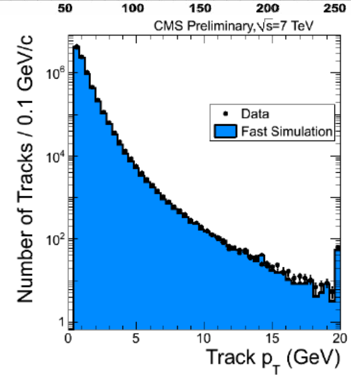
- Point-like approach to simulate material effects when crossing a layer
  - Several types of interactions consider
- Simplified reconstruction geometry
  - Connected cylindrical geometry + dea module map
  - Detailed magnetic field map
- SimHits -> RecHits
  - Smearing modules (CMSSW5)
  - Digitizers as in FullSim (CMSSW6)
- FastTracking
  - Seeding efficiency from MC truth
    - No fake tracks…
  - Track fitting, selection as in real tracking
- Performing FastSim + FastTracking in the same job



FullSim vs. FastSim tomography

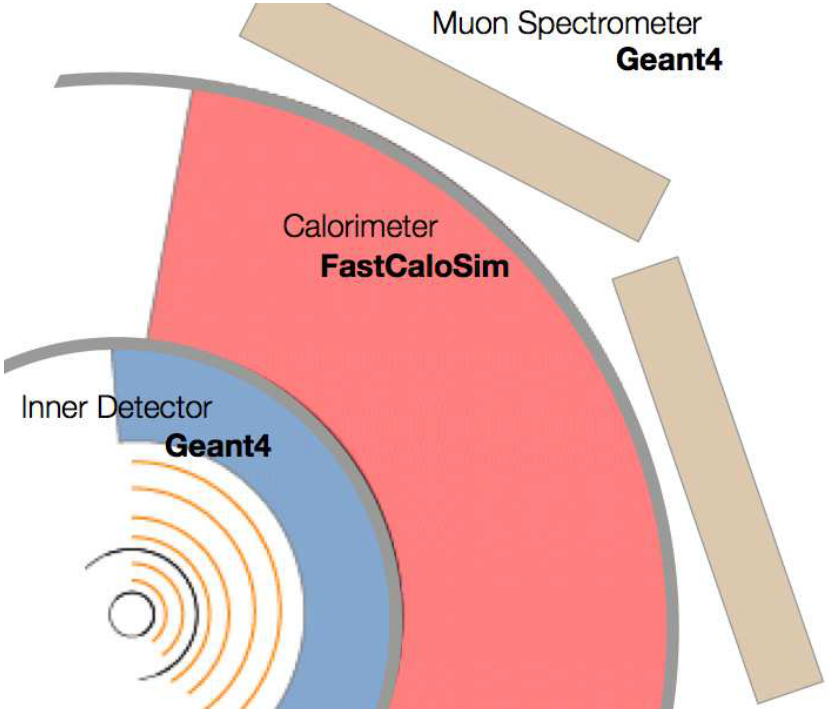






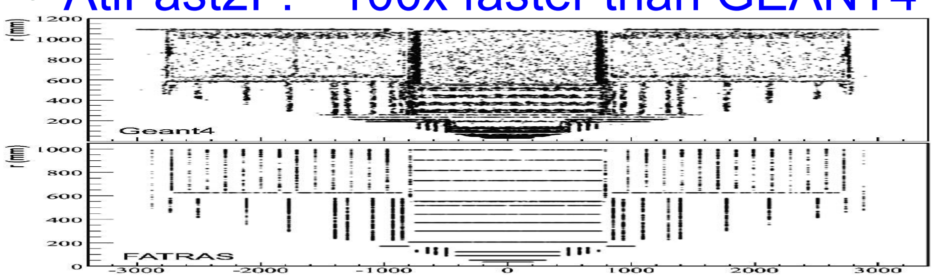
AtlFast II (F)

- Uses parameterization of calorimeter cell response to duplicate longitudinal and lateral energy profiles using a fine E/η grid
  - High momentum $e^{\pm}$, γ, $\pi^{\pm}$
  - Particle energy response, energy fractions in calo layers with fluctuations and correlations, average lateral shape
  - Lateral shower shape fluctuations, particle decays and leakage to muon spectrometer – planned to be addressed for Run2
- Combined with full GEANT4 for the rest
  - Including Frozen Showers for low energies
- AtlFast II : ~20x faster than GEANT4
- AtlFast2F: ~100x faster than GEANT4



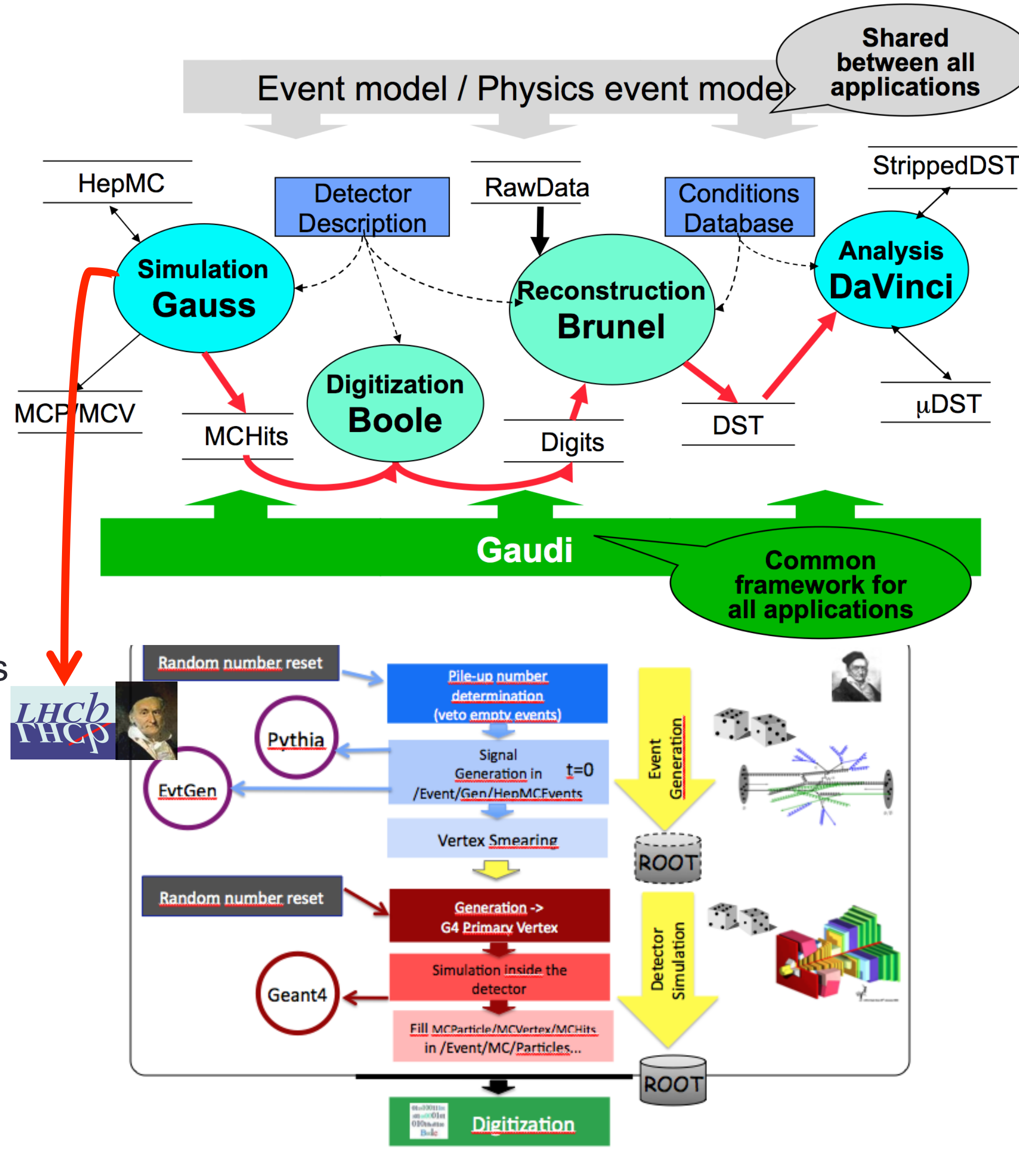
Sample	Full G4	Fast G4	Atlfast2
Minimum bias	551	246	31.2
$t\bar{t}$	1990	757	101
Jets	2640	832	93.6
Photons + jets	2850	639	71.4
$W^{\pm} \rightarrow e^{\pm}\nu_e$	1150	447	55.1
$W^{\pm} \rightarrow \mu^{\pm}\nu_\mu$	1030	438	57.0
Heavy ion	56k	21.7k	3050

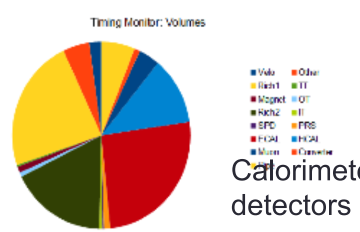
Simulation times in kSI2K seconds



LHCb simulation

- No fast simulation for transport so far
  - Most physics require full simulation
  - Several performance enhancements
    - transport cuts in calorimeters
    - low energy background parameterizations for muons system
- Work in progress to implement fast simulations
  - Available simulations are limiting some analysis
  - FullSim for signal only
    - 66 sec/event -> 3 sec/event
  - Not simulating CALO for out-of-time events
  - Generic samples for trigger studies and specific background determination
    - Fast MC would allow generating enough for some physics analysis

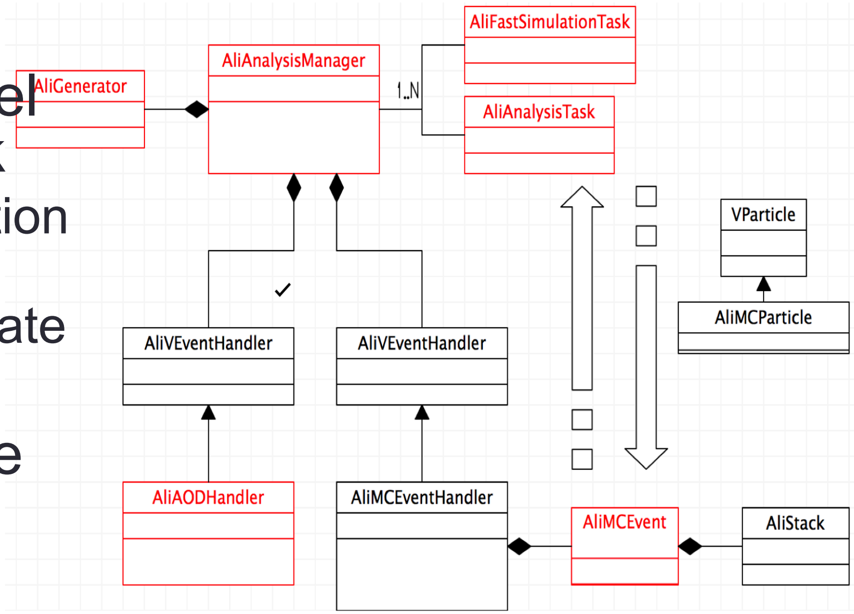


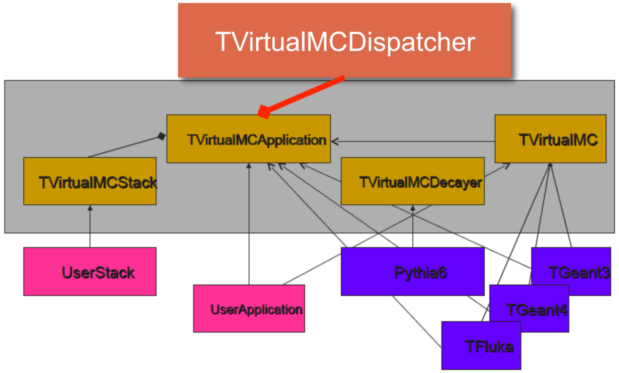


Calorimeters + RICH detectors ~80%

ALICE: Fast simulation as "analysis"

- Possibility to integrate fast parameterizations at generator level with the ALICE analysis framework
  - Analysis task becoming a fast simulation task
  - Acess to full kinematics, possibly update
  - Output directly AOD objects
- ALICE is using VMC to abstract the slow simulation engine
  - Code independent on MC
  - Extension for fast simulation being brainstormed (using a virtual dispatcher)
  - Would allow combinations of full/fast
    - Full after Fast: AF module processing kinematics event and injecting tracks into FullSim stack
    - Fast after full: Tracks stopped by the dispatcher during full simulation and injected into FastSim modules





VMC possible extension to support FastSim

Frameworks/extensions: ATLAS VFS

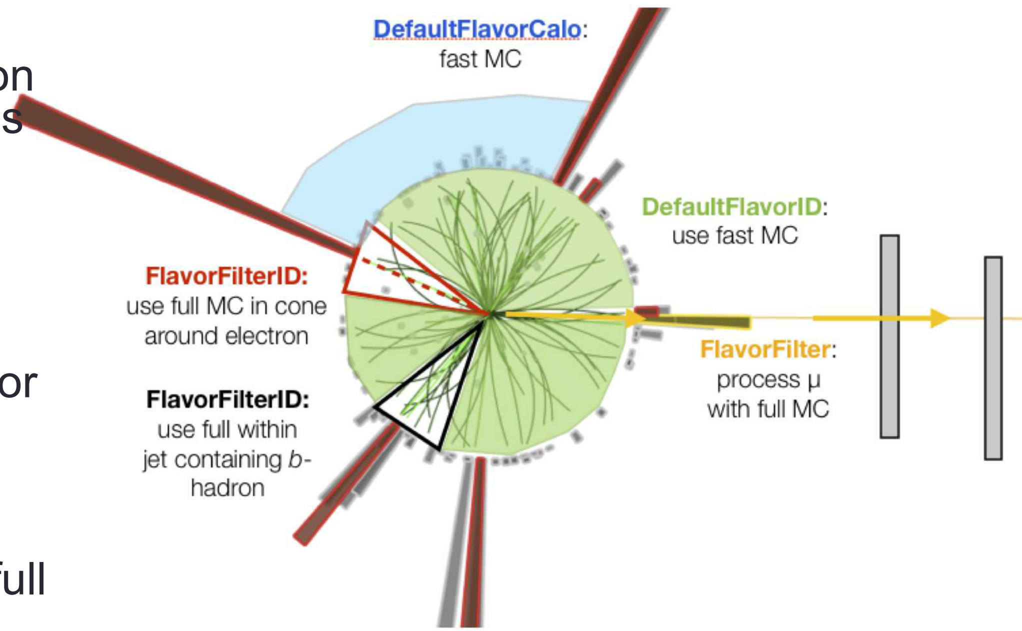
- VFS = Very Fast Simulation
  - Fast simulation, Fast digitisation and Fast reconstruction in the same job
  - No intermediate files, no extra job overhead
  - Performance-oriented solution: estimated 5-10 sec/event
    - Getting dxAODs or D3PDs for analysis in one go
- Digitization time dominated by ID, scaling linearly with pile-up
  - Fast Pile-Up: using in-time PU to model out-of-time PU using detector weights (1.4x for TRT, 2x for calorimeter)
- Reconstruction dominated by pattern recognition, track seeding and ambiguity treatment
  - Seeding using truth information

Can be removed or combined with the ATLAS slide

From VFS to ISF

- Mixing full and fast simulation within each physics event
  - For most analyses, high precision is needed only for some particles and regions
- Technically intermix with GEANT4 by controlling its stack of particles
  - Filtering geometrically by detector module
  - Filtering by particle flavor
  - Sending particle to the appropriate simulation service, full or fast
- All the benefits of an integrated approach
  - Reduced I/O, faster coupling
  - With an extra advantage in generality

The nutshell ISF vision

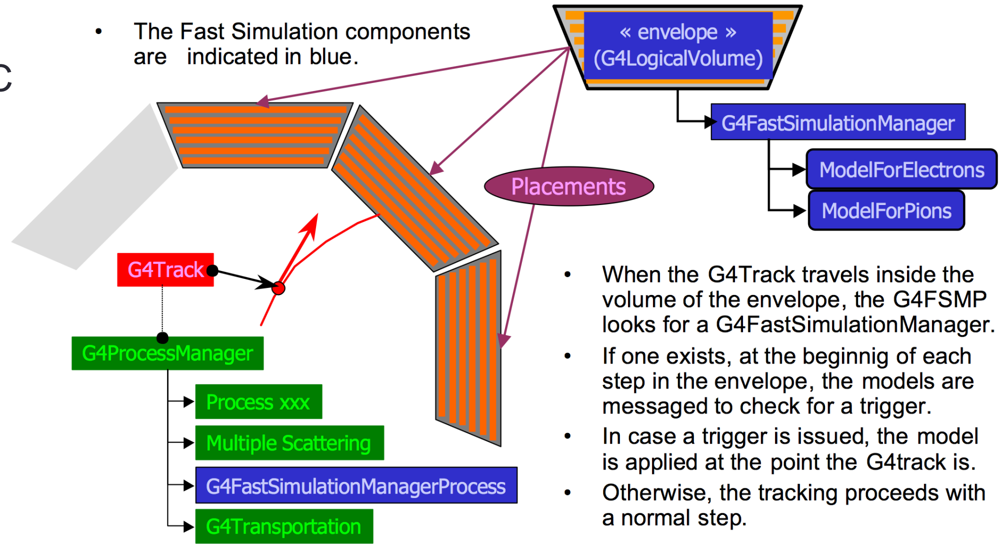


Can this generality be pushed upstream into the ~~slow~~ simulation engine?

GEANT4 & FastSim

M. Verderi – GEANT4 User's workshop, SLAC 2002

- GEANT4 HAS a mechanism to allow using user-defined fast simulation models per "envelope" since like ever
  - Looks to provide all elements to crossbreed Fast and Full simulations in a framework



- CMS is uses this concept of envelope for their GFlash custom parameterization
- ATLAS implemented handover mechanisms from full to fast more flexible than region-based
  - FastCaloSim triggered by pions in the outer part of the inner tracker
  - Regions of interest: cone arround the interesting particle
- Feeding this experience back into GEANT4 is important

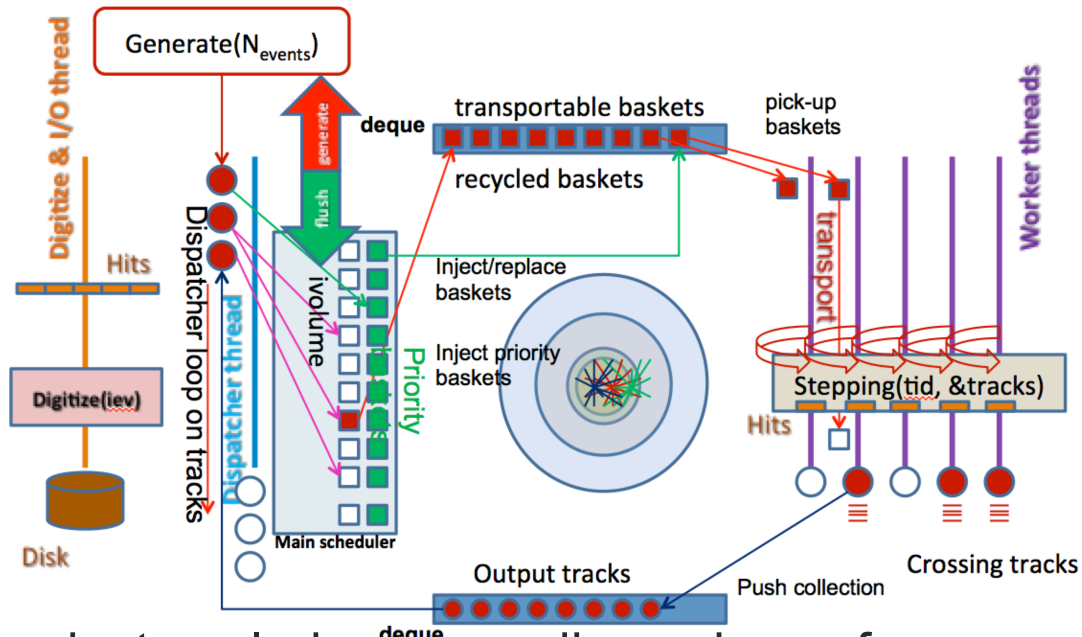
Summary I

- CPU in Run1 was dominated by MC productions
  - GEANT4 physics robustness was a major ingredient for the success
  - Many physics analyses in Run1 limited by the available MC statistics
  - Fast simulation was an important booster for simulated samples
    - It will become indispensable with the LHC increase in luminosity and pile-up
- LHC experiments exploring a wide range of FastSim approaches
  - Trying constantly to push up the performance limits (better and faster)
- The LHC upgrade challenges call for major changes in the FastSim frameworks: experiments working hard on that!
  - Understand very high PU impact and find solutions
  - Going from “it serves its purpose” to “integration” approach
  - Combining simulation, tracking, digitization to get ready to analyze data samples
    - Save intermediate steps, I/O
- Improving fast simulation performance does not make life easier…
  - Digitization and tracking become bottlenecks and demand their “fast” versions
- Fast and full simulation are NOT mutually exclusive
  - Performance comes from combining their features

Part 2: GeantV and challenges

The project goal

- Started in 2012 as a prototype for transport simulation
  - Multithreaded fine grained parallelism
  - Groups of tracks from many events (baskets) having geometry locality



- 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
  - Physics algorithm improvements

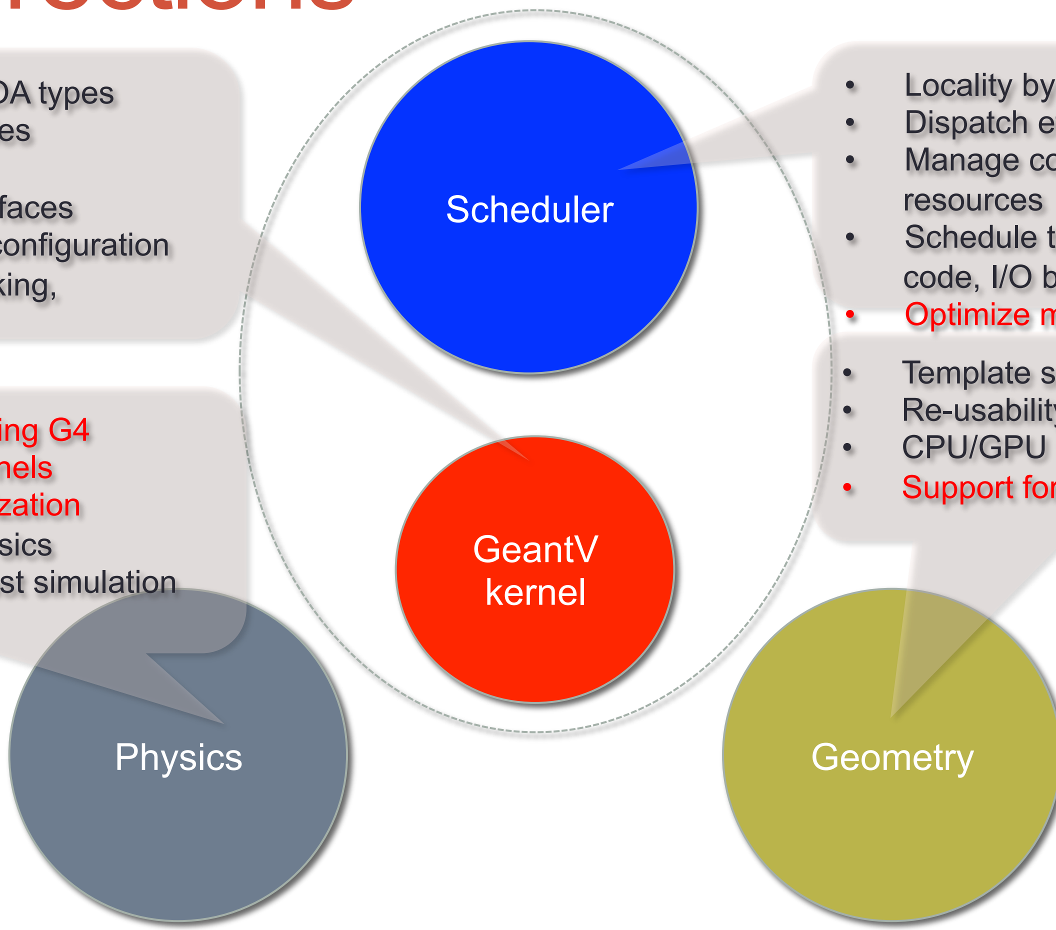
R&D directions

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

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

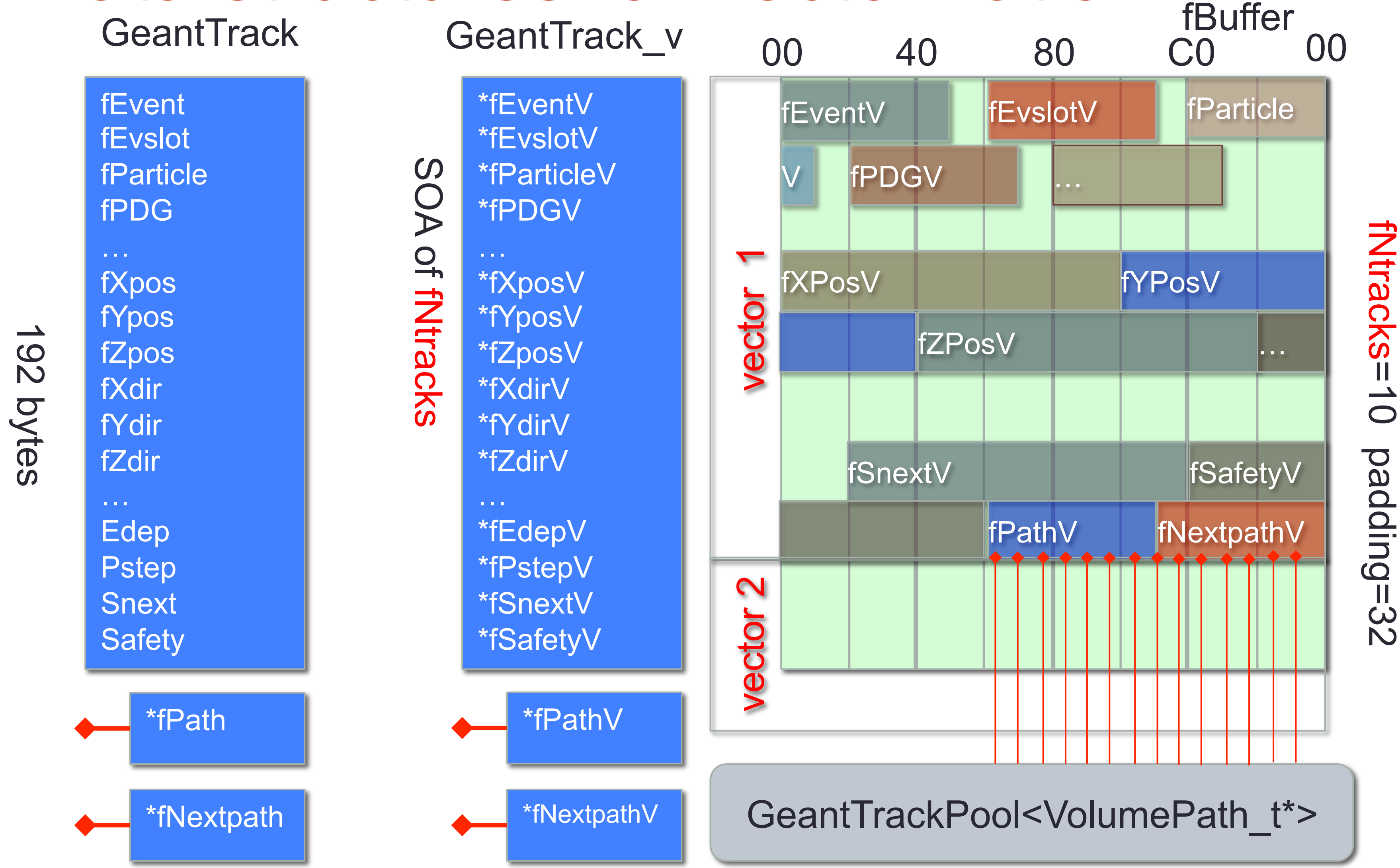
- Locality by geometry or physics
- Dispatch efficient vectors
- Manage concurrency and resources
- Schedule transportation, user code, I/O based on queues
- Optimize model parameters

- Template specialized algorithms
- Re-usability of kernels
- CPU/GPU transparent support
- Support for vectorization

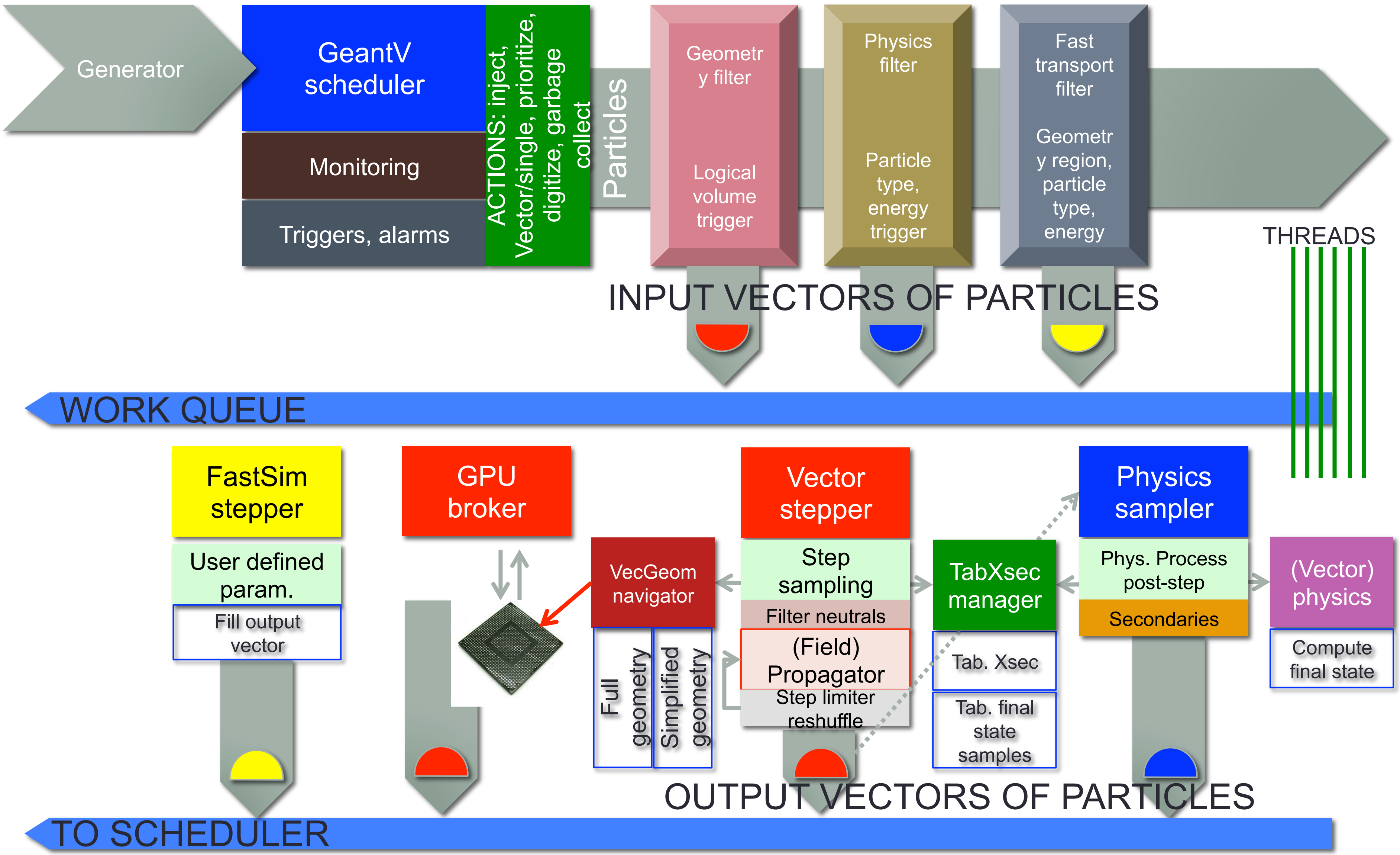


geant.web.cern.ch

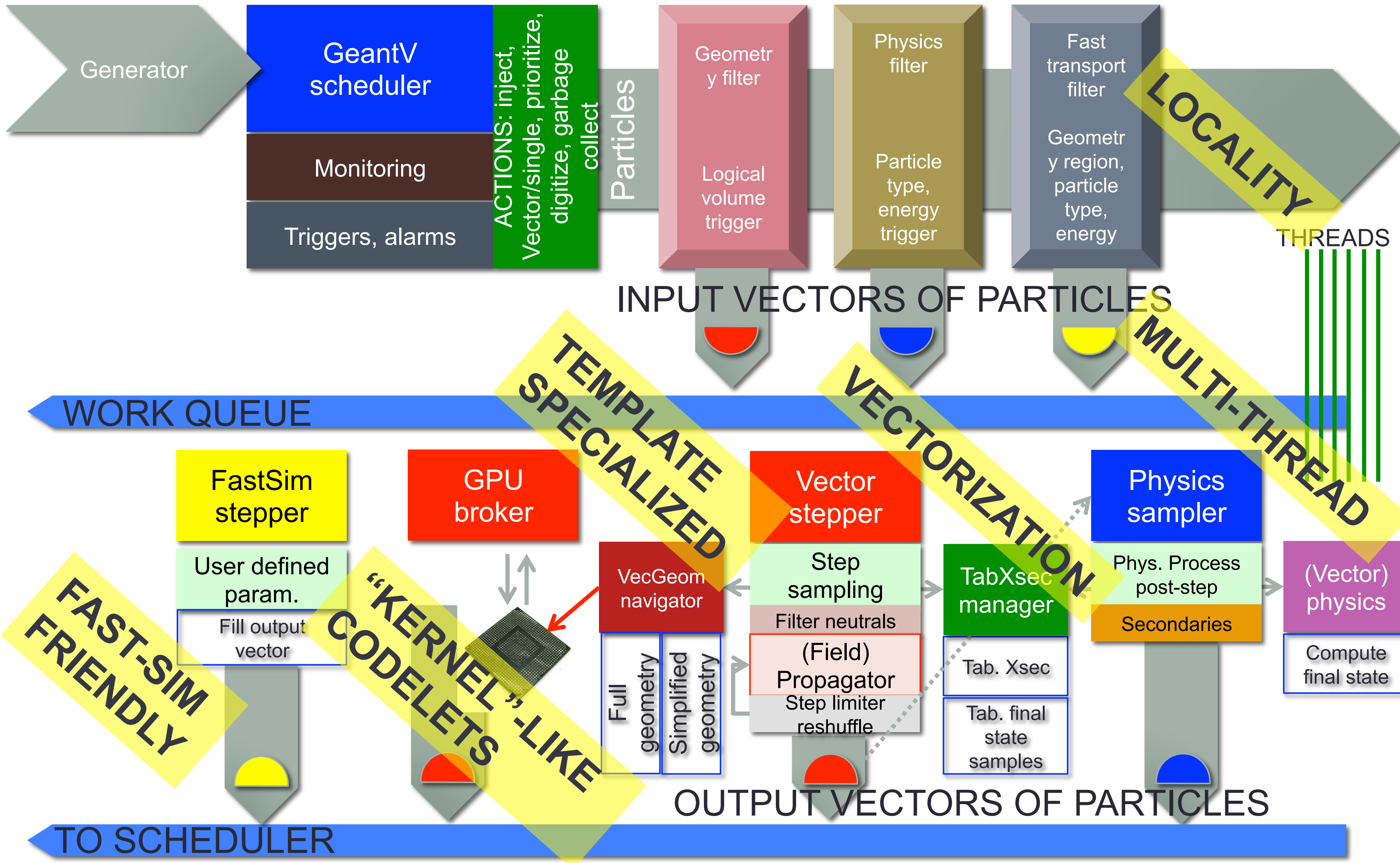
Data structures for vectorization



GeantV features and fast sim

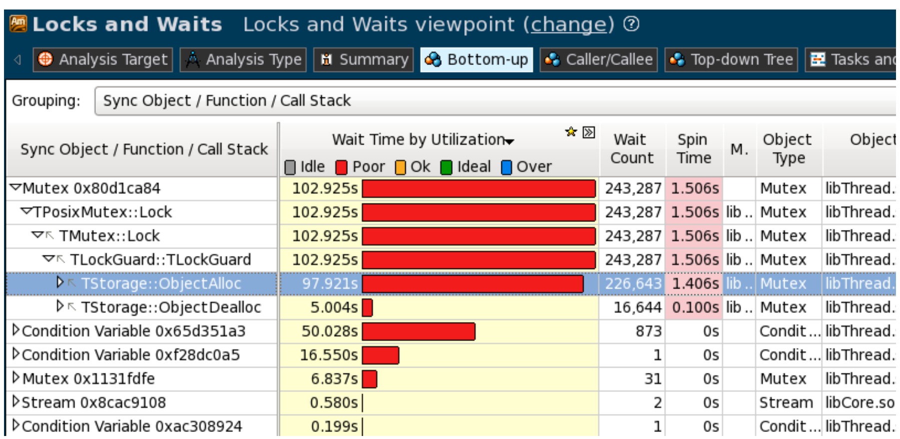


GeantV features and fast sim



Scalability for MT is challenging

- Performance is constantly monitored
  - Jenkins module run daily
- Allows detecting and fixing bottlenecks



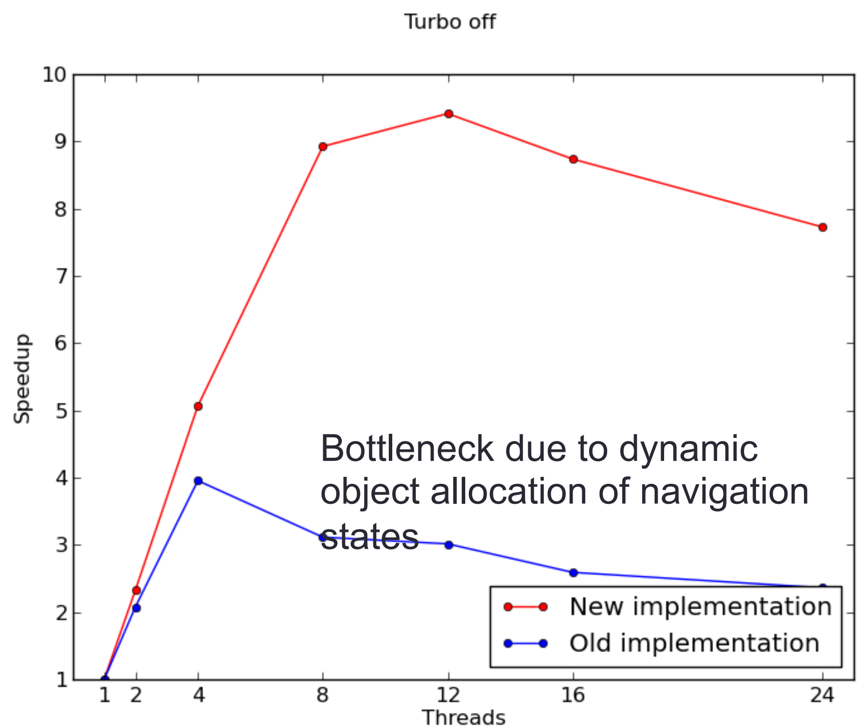
Locks and Waits — Locks and Waits viewpoint (change)

Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and

Grouping: Sync Object / Function / Call Stack

Sync Object / Function / Call Stack	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Wait Count	Spin Time	M.	Object Type	Object
Mutex 0x80d1ca84	102.925s	243,287	1.506s		Mutex	libThread.
TPosixMutex::Lock	102.925s	243,287	1.506s	lib..	Mutex	libThread.
TMutex::Lock	102.925s	243,287	1.506s	lib..	Mutex	libThread.
TLockGuard::TLockGuard	102.925s	243,287	1.506s	lib..	Mutex	libThread.
TStorage::ObjectAlloc	97.921s	226,643	1.406s	lib..	Mutex	libThread.
TStorage::ObjectDealloc	5.004s	16,644	0.100s	lib..	Mutex	libThread.
Condition Variable 0x65d351a3	50.028s	873	0s		Condit...	libThread.
Condition Variable 0xf28dc0a5	16.550s	1	0s		Condit...	libThread.
Mutex 0x1131fdfe	6.837s	31	0s		Mutex	libThread.
Stream 0x8cac9108	0.580s	2	0s		Stream	libCore.so
Condition Variable 0xac308924	0.199s	1	0s		Condit...	libThread.

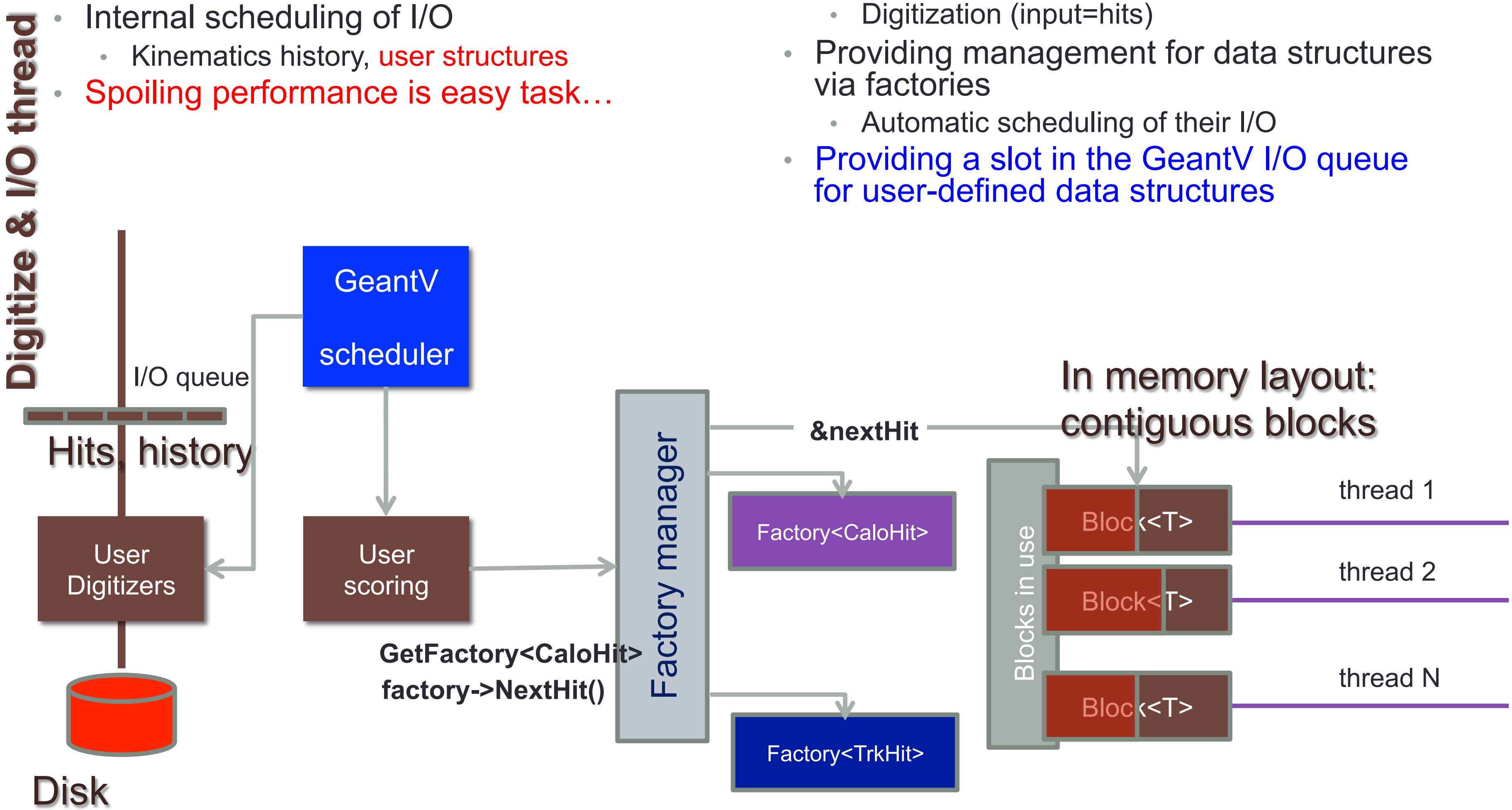
1000 events with 100 tracks each, measured on a 24-core dual socket E5-2695 v2 @ 2.40GHz (IVB).



Exposing optimizations to user code

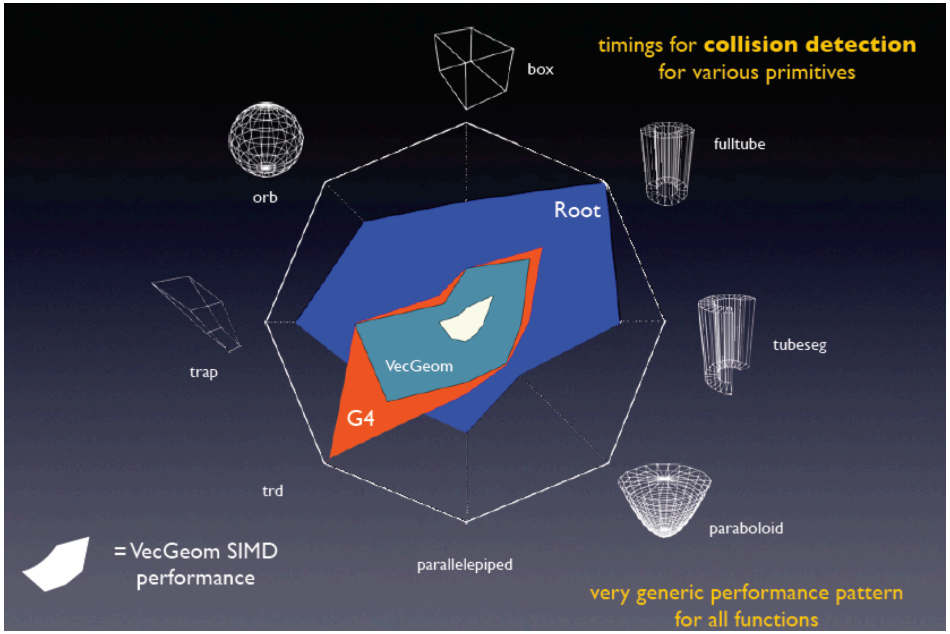
- GeantV works with vectorized stepper
  - Vectorized callbacks to user code
- Internal scheduling of I/O
  - Kinematics history, user structures
- Spoiling performance is easy task…

- Supply aligned containers to user code
  - Hit production (input=tracks)
  - Digitization (input=hits)
- Providing management for data structures via factories
  - Automatic scheduling of their I/O
- Providing a slot in the GeantV I/O queue for user-defined data structures



VecGeom – optimizing simulation geometry

- Building a high performance multipurpose geometry library
    - Single/multi particle SIMD
    - Generic code for CPU/GPU
    - R&D a generic multi-platform programming approach
- Major project integrated with the GeantV prototype



- Link to Sandro's talk here

"Fast" physics and upgrades

- Optimizing the performance of GEANT4 physics will take a log time
  - Vectorization, kernels + algorithms review from these perspectives
- **Goal:** have a compact an simple form of realistic physics to study the prototype concepts and behavior
- **Requirements:** mimic the most important effects of the "real physics" to the tracks and to the characteristics of the transport
  - 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
  - all major processes are involved
  - flexible number of final states for all particles, all active reactions, all elements are also extracted from GEANT4 and stored in tables

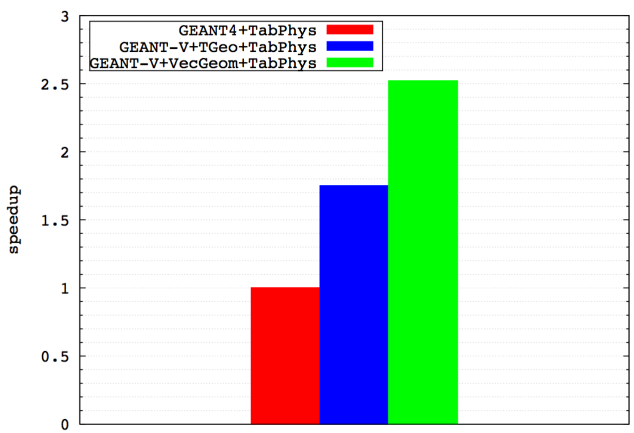
Status:

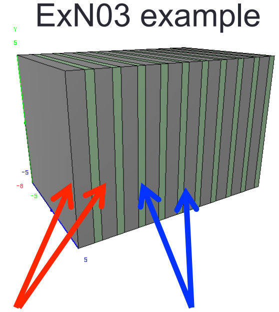
- a complete particle transport (except msc) has been implemented based on these tables both behind the prototype and behind GEANT4
- possible to test new concepts, performance relative to GEANT4 tracking
- individual physics processes can be replaced by their optimized version when ready

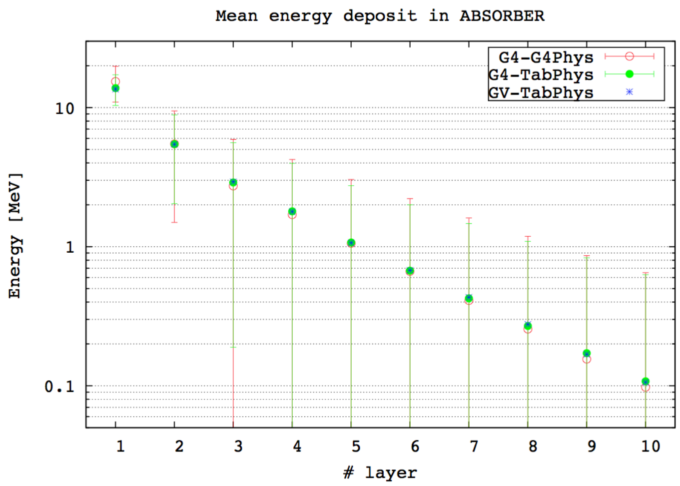
GEANT4 Geant4 physics	GEANT4 TabXsec physics
GeantV TabXsec physics	GeantV Optimized physics

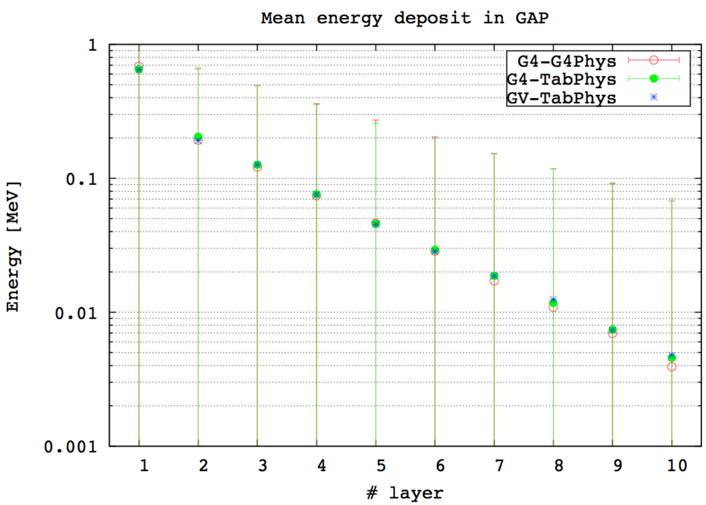
Preliminary performance checks

- 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
- No energy dependence of performance gain
  - Extension to (simple version of) CMS geometry soon possible



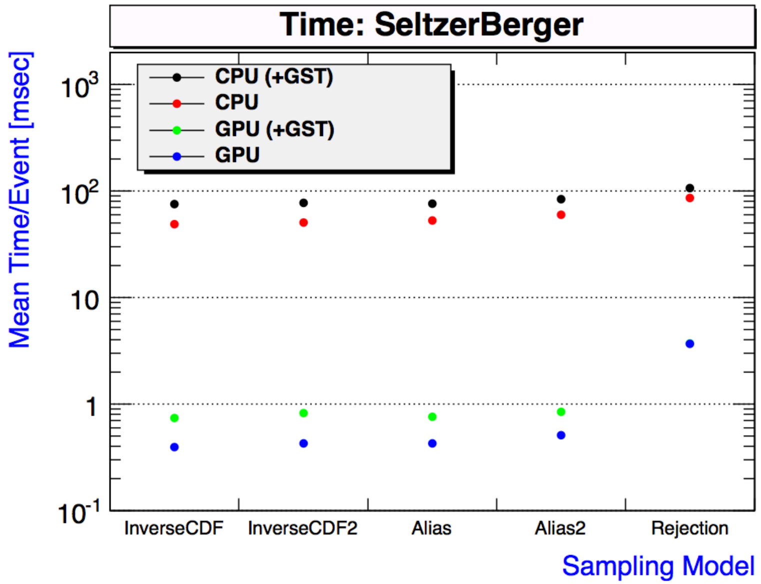


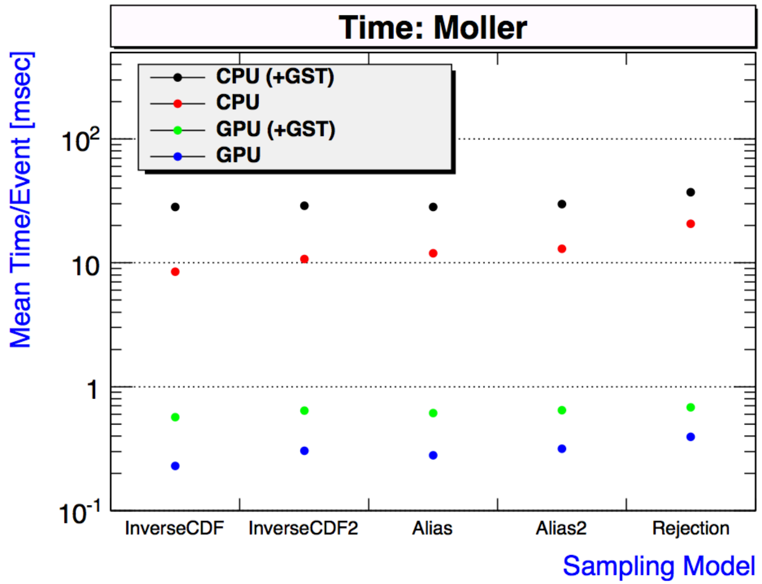




Techniques for optimizing physics

- Sampling secondary particles produced by high energy physics interactions
  - minimize conditional branches and non-deterministic implicit loops
  - replace the conventional "Composition and Rejection Monte Carlo methods"
  - use vectorizable inverse transformation (inverse CDF) or alias methods
- Performance on Intel Xeon (X5650) and NVidia Tesla GPU (Kepler 20M)





Summary II

- We need a qualitative jump in the way our simulation SW uses the hardware
  - Small window of time to R&D new approaches
  - Looking at performance from all angles, including fast simulation
- High performance is within reach
  - Vectorization and locality can give the expected results
  - Extending to GPU is a must
  - Optimizing geometry and physics is a long scale effort
- Including direct support for fast simulation in the transport framework is mandatory
  - No limits, at the extreme approach GeantV should be usable as a fast simulation framework
  - A lot to learn from the existing “integration” approaches
- “Slow” and “fast” have to crossbreed!

Acknowledgements

- Andreas Salzburger, Andrea Giammanco, Lukas Vanelderen, Vladimir Ivantchenko, Gloria Corti, Marco Cattaneo, Andreas Morsch, Marc Verderi

- All people contributing with material and/or giving input for this talk

Thank you!