Performance results of the GeantV prototype with complete EM physics

Andrei Gheata for the GeantV R&D team

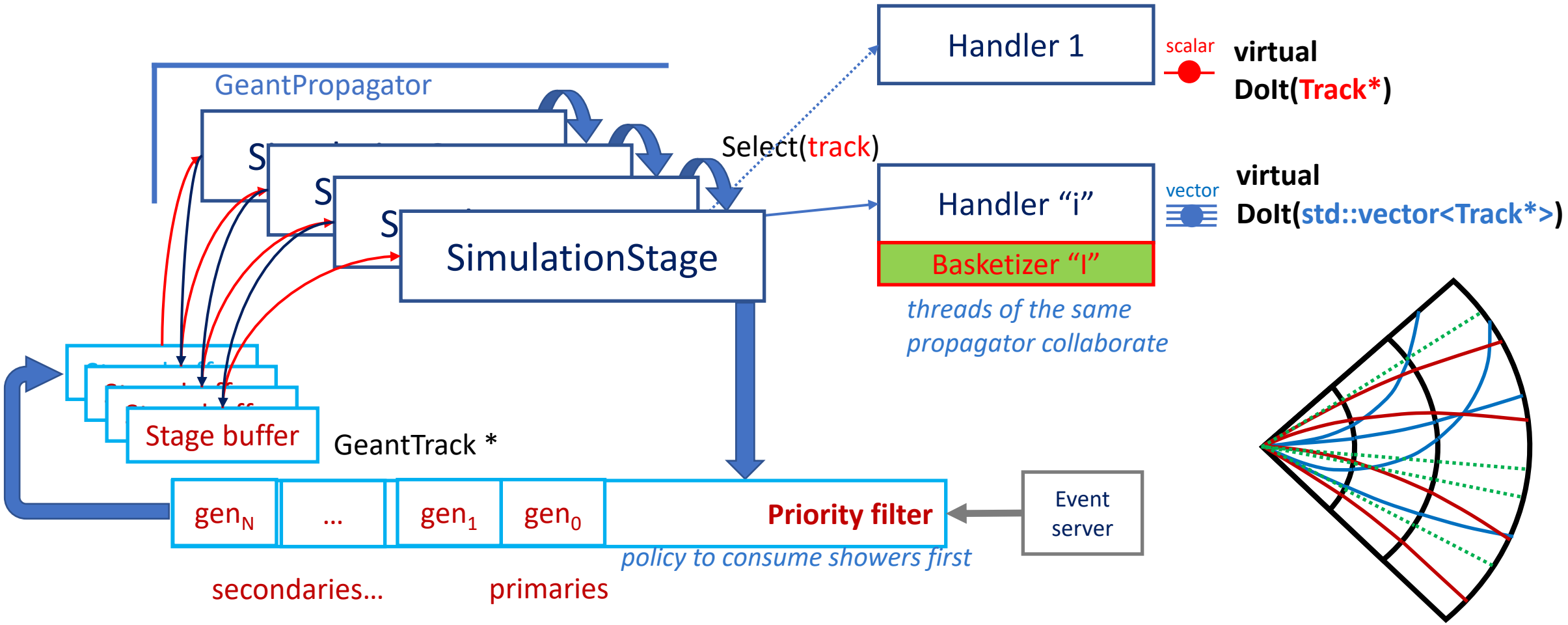
ACAT 2019, Saas Fee

Simulation performance R&D

- GeantV: a general performance study for the full simulation workflow
- How to improve the steering framework
  - Track-level parallelism, basket rather than single track workflow
  - Improving instruction and data locality
  - Leveraging vectorization techniques (VecCore)
  - Adaptable to new hardware and accelerators
- How to improve individual simulation components
  - VecGeom: new geometry modeler using efficient template programming to handle single/multi particle queries
  - New physics framework, more simple and efficient, vectorization-aware
  - VecMath: new SIMD-aware RNG and math algorithms

GeantV multi-particle processing

Both scalar/vector flow are supported

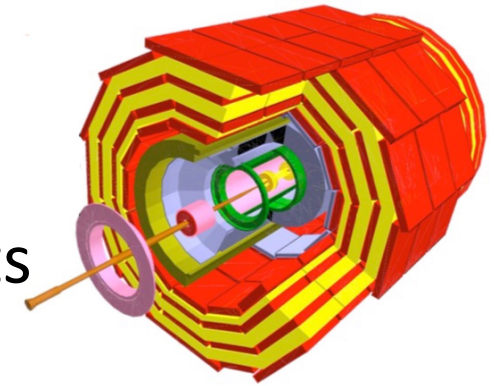


Where are we today?

- Full EM shower simulation with full complexity
  - Realistic geometry and physics configuration
    - Possible to run full EM simulation in the context of a LHC experiment
  - User interfaces scoring efficiently in multi-event multi-threaded environment
    - Ongoing integration exercise with CMSSW by CMS
  - First demonstrator for RNG reproducibility
  - A set of pre-beta tags available
- Thorough ongoing performance study
  - Detailed comparisons: different GeantV modes and Geant4
    - Hotspots, performance counters, MT
- Preliminary set of conclusions including:
  - Vectorization and locality: benefits and limitations
  - Current limits for multi-threading in basketizing environments

What we compare

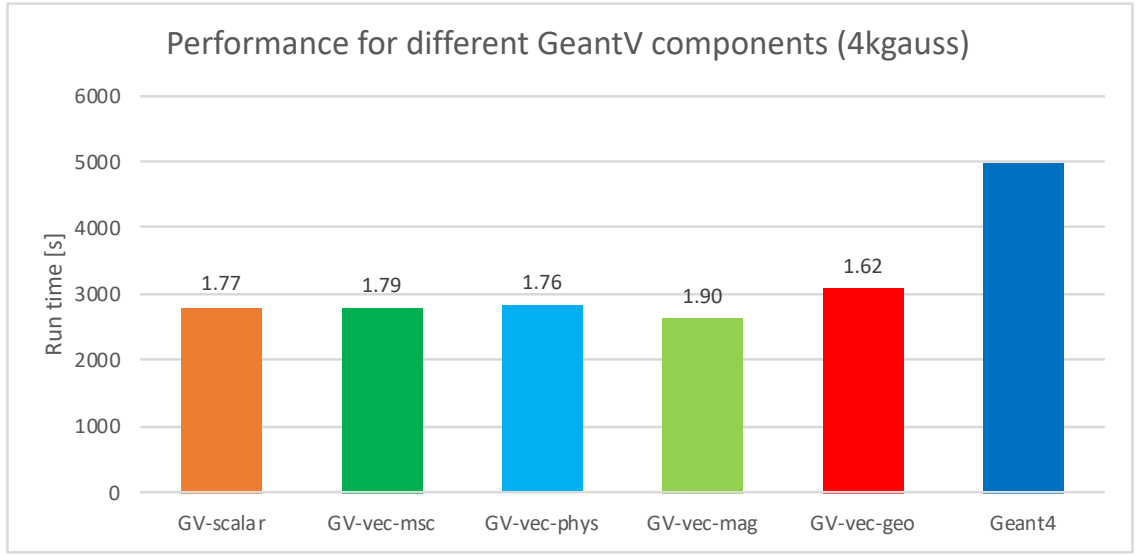
- Examples: simplified sampling calorimeter and a full CMS simulation
- GeantV: several configurations
  - Basketization ON/OFF for different components: geometry, final state sampling (physics), field propagation, MSC
  - Field ON/OFF
  - Single track mode ON/OFF
  - Basket dispatching to vectorized vs. scalar code
- Geant4: equivalent physics list, geometry setup and cuts
  - Single threaded or MT mode

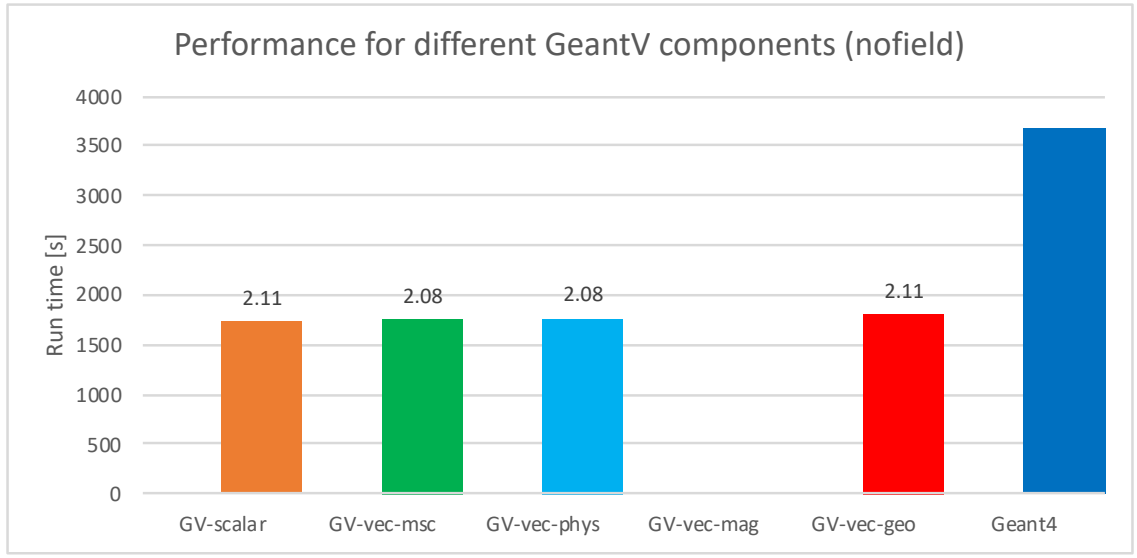


General performance FullCMS

Machine specs here

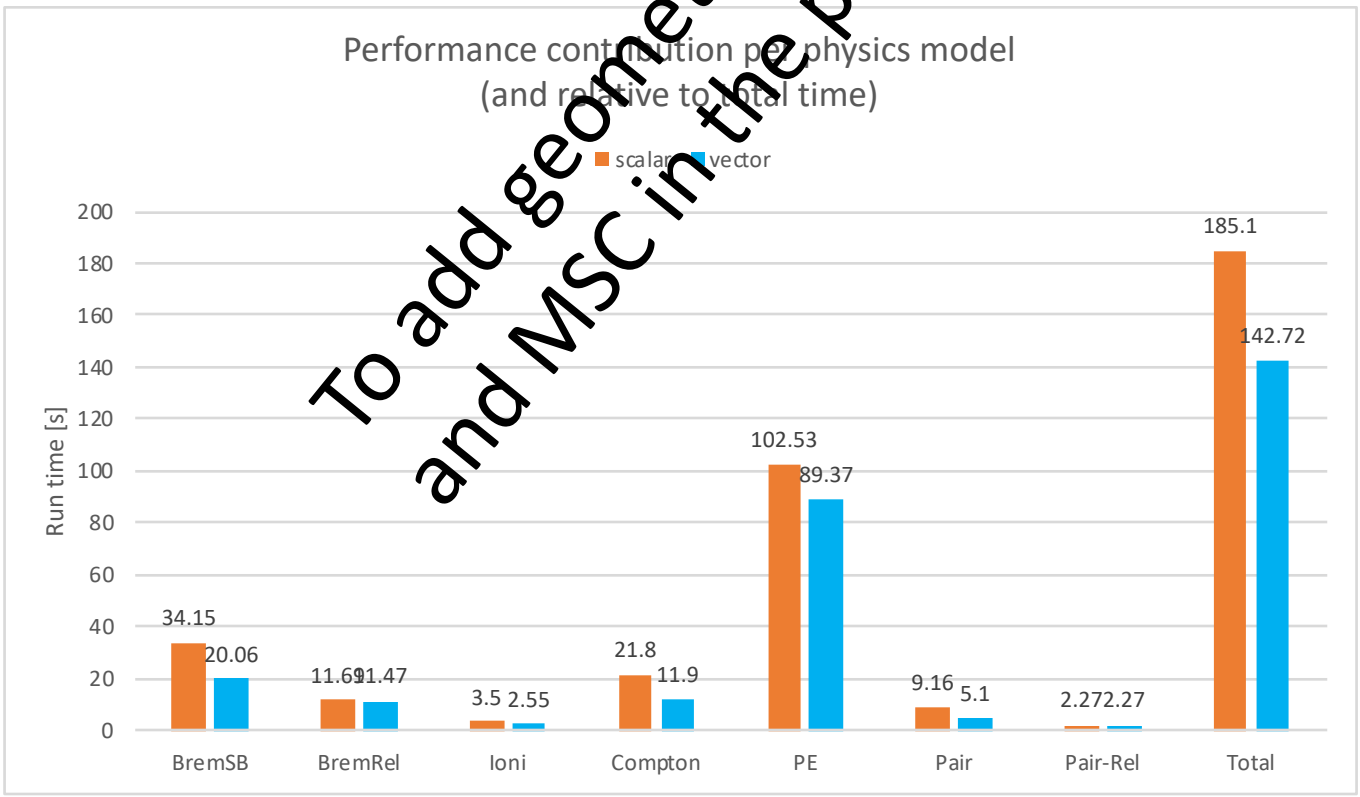
- Overall performance improvements of 1.9 – 2.1 for the most efficient configurations
- Gains from vectorization benefits visible for just a subset of stages (up to ~15% overall)
- Benefits from stage-local workflow visible only for up to 15%, dependent on machine topology
- The rest of performance gain coming from other improvements/ code simplification (see next)





Vectorization performance

- Intrinsic vectorization gains per model visible in profiling
  - Lower that the speed-ups in (ideal) unit tests (1.3-2.5)
  - Overall intrinsic gain of only 1.3, lower than data gathering/scattering loss!
- Important gains for some methods
  - Field propagation: x%, multiple scattering angle/shift sampling y%
- Performance loss in case of “small” hotspots (e.g geometry volumes)
- Basketizing is efficient only when applied to “dense FLOP” algorithms



To add geometry, field and MSC in the plot

A more detailed performance view

- Implemented a special “single track” mode
  - GeantV workflow using stack-like approach, transporting single tracks through all stages
- Different levels of performance loss in single track mode
  - From 0 to ~15-20%, depending on the test machine
  - Still not understood what contributes to results being different
  - Next step: single track mode only for selected stages
- Performance indicators give already valuable (but not comprehensive) hints on locality

	CPU time	G4/GV	single/default	FPC	IPC	FMO	TLB_DM G4/GV	TLB_IM G4/GV	L1_DCM G4/GV	L1_ICM G4/GV
GV-4kgauss	2827	**1.76**		0.26	1.06	0.56	0.74	11.16	1.38	7.63
GV-4kgauss-strk	3270	**1.52**	**1.16**	0.21	1.05	0.47	0.4	7.72	1.33	2.04
G4-4kgauss	4987			0.13	0.8	0.33				
GV-nofield	1754	**2.09**		0.25	1.1	0.51	0.71	24.97	1.28	16.65
GV-strk-nofield	2031	**1.81**	**1.16**	0.21	1.15	0.41	0.6	38.72	1.39	3
G4-nofield	3668			0.13	0.85	0.32				

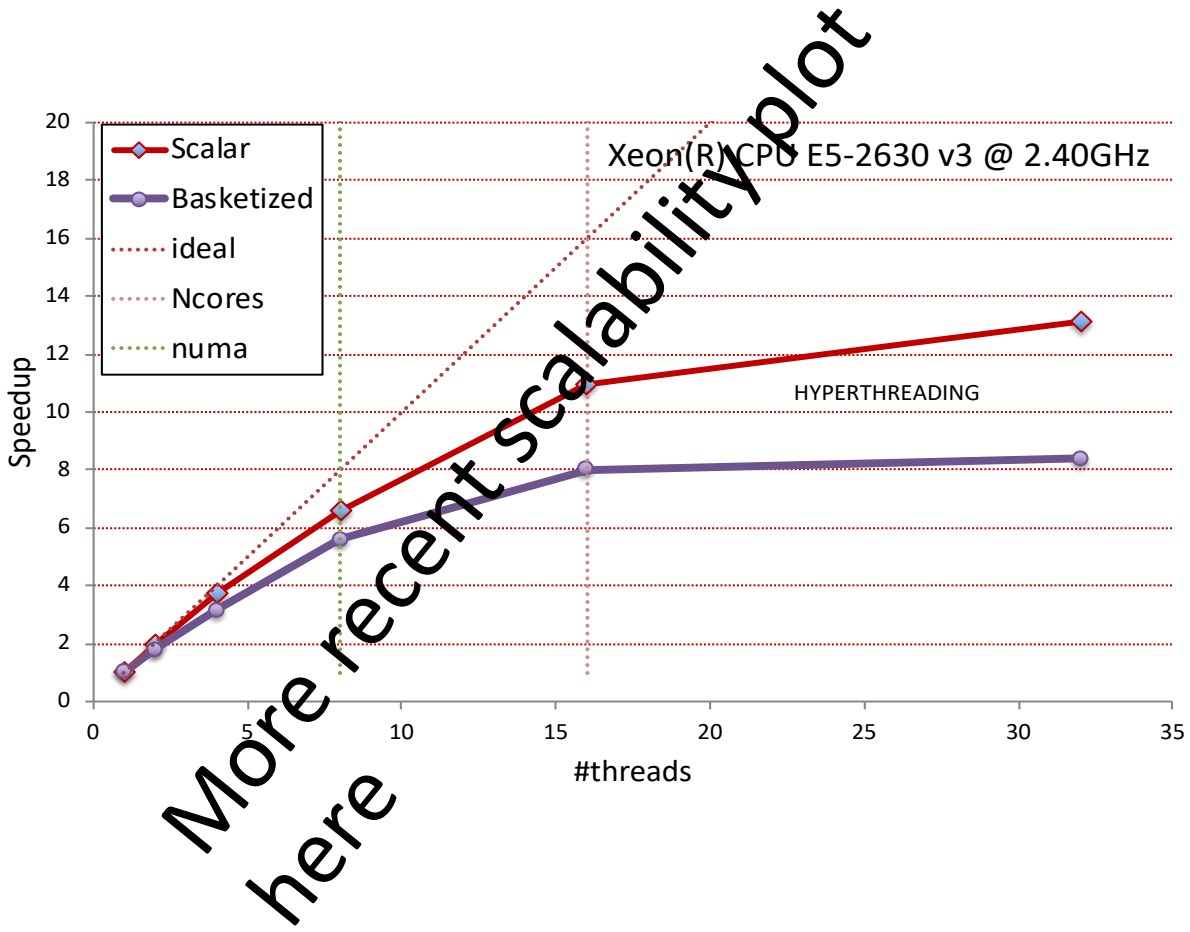
Preliminary conclusions for single thread performance

- GeantV executes much less instructions for the same number of FLOP
  - Much less TLB/L1 cache misses (to be quantified in %time)
  - Implicit better numbers for instructions per cycle (IPC), computation intensity, CPU utilization
- The gains from locality (~15%) and vectorization (~15%) explain only a small part of the factor of ~2x speedup
  - Simplified/more efficient code, library size, less deep call stack and less virtual calls – just some of the possible reasons
  - Quantifying these effects is very important
- The limits of applicability of the GeantV “basket” model now visible
  - Reasonably “hot” computation hotspots show clear benefits

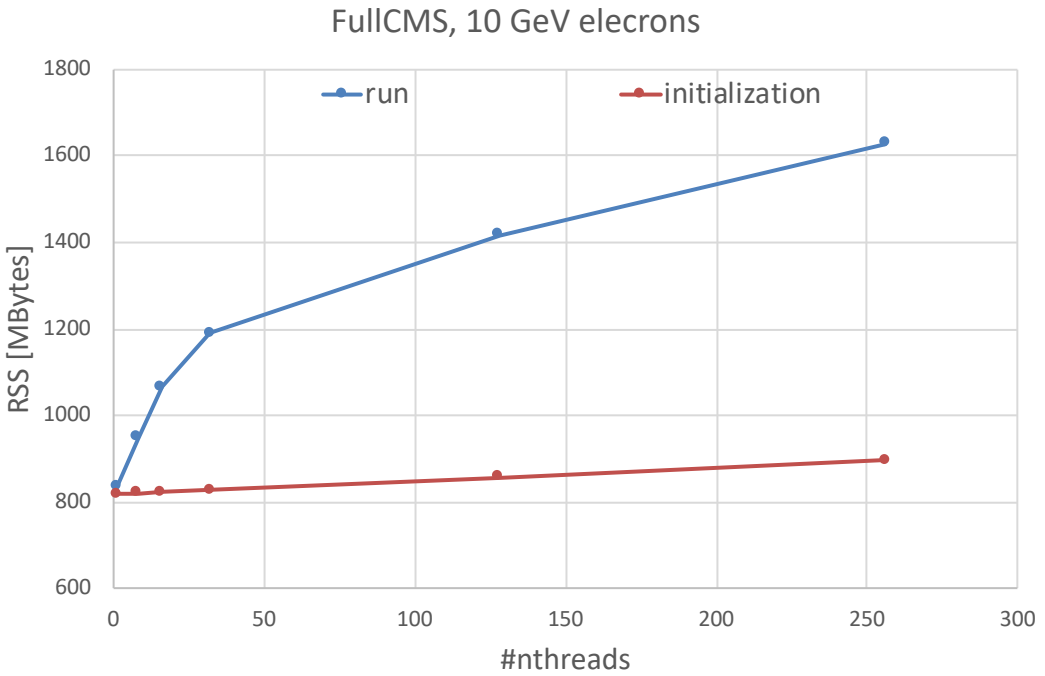
Multi-threading performance

- Track workload shared among threads
  - Event tails spread across threads require frequent flushing (less efficient)
  - Flushing baskets de-balances the work
- Several improvements to reduce Amdahl, partially successful
  - Work stealing queues, memory contention reduced
- A compromise requires less track sharing at the expense of more memory used
  - Sets of events owned by or having affinity to threads
  - Different degrees of event affinity for threads under investigation

Current MT performance



More recent scalability plot here



- Memory footprint higher than G4 (~3x)
- Size of data structures just after initialization is the major contributor
- Rather good behavior with increasing #threads

Short-term work plan

- Continue the performance analysis
  - The goal is to quantify where the bulk of the gain (~60-70%) is coming from
  - Consolidate and complete the performance numbers
    - Dependence on the architecture and machine features
    - Conversion of cache misses into absolute/relative time
  - More detailed locality study
    - Single track mode for selected stages only
  - Final fixes and consolidations for the beta release (now at pre-beta4)
- Understand the most profitable directions to work on to improve Geant4
  - Code simplification: physics framework and step management
  - Areas where locality can be improved, by adopting a basket-like workflow

Where we go from here?

- GeantV R&D at a stage where we can measure/evaluate
  - The actual benefits of the main idea and of the side developments
  - Things that work and things that don’t
- Preparing a detailed technical document on our study
  - Facts, numbers and lessons learned
  - A hopefully useful guide with performance hints for other applications
- Decide on the best strategy to follow to bring the benefits of this study in production