Performance of GeantV

Soon Yung Jun, Philippe Canal, Guilherme Lima

Sept. 13, 2019

PDS Geant R&D Retreat

This talk

- Performance of GeantV (the latest tag, pre-beta-7)
    - Benchmark: Geant4/GeantV and different configurations
    - SIMD Vectorization
    - Platform dependency
    - Other performance metrics (FPC, IPC, FMO, Cache misses)
    - Conclusion
- GeantV summary paper
    - Motivation and proposed time line
    - Status

Performance Benchmark: Tested Platforms

- Processor-Cores-CPU[GHz]-Memory[GB]-Cache[MB]-SIMD

Processor	Core	CPU	Mem	Cache	SIMD
Intel E2620 (Sandy Bridge)	2x6	2.0	32	15	AVX
Intel E2680 (Broadwell)	2x14	2.4	128	35	AVX2
AMD 6128 (Opteron)	4x8	2.3	64	15	SSE4

- Cache Size

Processor(*)	L1 set	L2 set	L3 set
AVX-2.0-15	6x32 KB 8-way	6x256 KB 8-way	15 MB 20-way
AVX2-2.4-35	4x32 KB 8-way	14x256 KB 8-way	35 MB 20-way
SSE4-2.3-15	8x64 KB 2-way	8x 512 KB 16-way	2x6 MB

* Processor Convention: SIMD-CPU[GHz]-Cache[MB]

Performance Comparison: Benchmark

- Benchmark: baseline
  - GeantV (pre-beta-7) vs. Geant4 (10.5)
  - The standalone Geant application using the 2018 CMS gdml (FullCMS/GeantV vs. full_cms/Geant4) with B=fieldMap
  - 10×10 GeV $e-$/event, 1000 events, 1-thread
  - measurements under quiet batch nodes (error $\ll 1\%$)
- CPU Time [sec]

Processor	Geant4	GeantV	GeantV-vec	G4/GV	G4/GV-vec
AVX-2.0-15	4938	2621	2331	1.88	2.12
AVX2-2.4-35	2182	1628	1530	1.34	1.43
SSE4-2.3-15	6627	4457	4333	1.49	1.53

  - Geant4/GeantV(scalar) performance widely varies: $\sim (1.3 - 1.9)$
  - marginal gain by SIMD vectorization: $(5 - 15)\%$
- Why is the gain by vectorization small?
- What are sources of performance difference between GeantV(scalar) and Geant4?

Performance Comparison: Magnetic Field

- Performance with different field configurations: ex. on AVX

Magnetic Field	GeantV [sec]	Geant4/GeantV	Geant4/GV-vec
Zero	1794	1.86	1.95
Uniform (3.8T)	2412	1.97	2.19
CMS Field Map	2621	1.88	2.12

  - Relative performance of Geant4/GeantV are reasonably stable

Vector Instruction and Gain in CPU

- % of Vectorization = (PAPI_DP_VEC)/(PAPI_DP_OPS)
  - PAPI_DP_OPS = Floating point (double precision) operations
  - PAPI_DP_VEC = Double precision vector/SIMD instructions
- Counters in [1 Billion]: ex. on AVX

Mode	PAPI_DP_OPS	PAPI_DP_VEC	% vectorization	CPU gain
scalar	1770	277	15.67	-
vec-geo	1771	333	18.82	0.96
vec-mag	1858	814	43.83	1.08
vec-msc	1789	397	22.24	1.02
vec-phys	1785	343	19.25	1.00
vec-all	1868	1051	56.26	1.00
vec-opt	1868	996	53.35	1.12

  - vec-opt = all vector modes are turned on except geometry
- % of vectorization is significant, but the overall gain is small
  - basketization overhead (not shown here): $\sim (10-25)\%$
  - inefficiency due to gather/scatter and mask operations

Scheduler and Locality

- Single track mode (GeantV-strk)
  - emulation of the Geant4-style tracking
  - a reference for a measure of the scheduler performance and data locality
- CPU Time in [sec] and their ratios on different platforms

Processor	GeantV	GeantV-strk	strk/default
AVX-2.0-15	2621	2960	1.13
AVX2-2.4-35	1628	1533	0.94
SSE4-2.3-15	4457	4817	1.08

  - Impact of the GeantV scheduler or data locality is not the primary source of performance difference between Geant4 and GeantV (scalar)

Performance Comparison: Platform dependency

- Performance variation (α) with respect to AVX-2.0-15 (Time0)
- α factor taking into account the clock speed

$$\alpha = \frac{\text{Time0} \times \text{CPU0}}{\text{Time} \times \text{CPU}} \tag{1}$$

- $\alpha > 1 (\alpha < 1)$: more (less) efficient than AVX

Processor	GeantV	GeantV-vec	Geant4
AVX-2.0-15	1	1.13	1.88
AVX2-2.4-35	1.34	1.26	1.97
SSE4-2.3-15	0.52	0.47	0.65

- Intel: Geant4 is more sensitive to the size of cache
- AMD: Both are significantly bad with respect to Intel

Performance Comparison: Geant4 Libraries

- Exclusive time (%) of big libraries

Library (%)	AVX	AVX2	SSE4
libGeant_v.so	42.1	46.3	43.2
libRealPhysics.so	36.0	34.2	37.3
libGeantExamplesRP.so	14.1	14.1	14.5
libc-2.12.so	3.8	1.8	1.1
libVmagfield.so	3.1	2.8	3.1
libm-2.12.so	0.6	0.6	0.6

- There are no much variations in the percent of time over different CPUs/Cache-Size
  - the performance difference is a global effect (i.e., not driven by a single module or a set of functions)

Performance Comparison: GeantV Libraries

- Exclusive time (%) of big libraries

Library (%)	AVX	AVX2	SSE4
libG4geometry.so	41.8	43.6	42.3
libG4processes.so	22.0	20.8	21.0
libG4global.so	7.3	8.0	7.5
libG4tracking.so	7.3	6.5	7.2
libG4track.so	6.0	4.7	5.8
full_cms	5.2	6.1	6.6
libG4clhep.so	3.3	3.0	3.0
libm-2.12.so	2.7	3.5	2.9
libG4particles.so	1.2	0.7	1.0
libG4digits_hits.so	1.1	1.3	1.0

- No significant variation either
  - the overal performance difference between GeantV (sequential) and GeantV is a global effect

Instruction/Cycle and FLOPS/Cycle

- Instruction(INS)/Cycle(CYC) = IPC
  - Good Balance with Minimal Stall
- INS/CYC in 1B counters

Processor	GV INS/CYC	GV IPC	G4 INS/CYC	G4 IPC
AVX-2.0-15	7038/6610	1.06	8388/10788	0.78
AVX2-2.4-35	6474/5521	1.19	8914/5514	1.62
SSE4-2.3-15	7813/8839	0.88	8459/11228	0.75

  - INS: Instruction completed
  - CYC: Total Cycle
  - Geant4: The total number of instructions is nearly constant, but cycles varies significantly
  - GeantV: IPC is more stable across different platforms
- FPC = FLOPS/Cycle: CPU Utilization
  - FLOPS: Floating point operations (Single/Double Precision)
  - FPC follows similar behaviors to IPC (INC $\propto$ FLOPS)

Performance Comparison: L1/L2 Cache Miss

- L1 Cache Miss : in 1B counters

Processor	GV (ICM)	G4(ICM)	GV (DCM)	G4(DCM)
AVX-2.0-15	54	429	218	269
AVX2-2.4-35	39	511	188	272
SSE4-2.3-15	49	309	141	144

  - ICM/DCM: Instruction/Data Cache Miss
  - Level 1 latency = 3 cycles
  - GeantV shows much significantly less ICM
- L2 Cache Miss : in 1B counters

Processor	GV (ICM)	G4(ICM)	GV (DCM)	G4(DCM)
AVX-2.0-15	19	36	86	46
AVX2-2.4-35	23	29	101	51
SSE4-2.3-15	17	3.6	55	10

  - Level 2 latency = 12 cycles
  - Intel: GeantV has less ICM and Geant4 has less DCM
  - AMD: opposite to Intel

Performance Comparison: L3 Cache

- L3 Cache Miss : in 1B counters

Processor	GV (TCM)	G4(TCM)	GV (TCA)	G4(TCA)
AVX-2.0-15	1.9	0.19	109	80
AVX2-2.4-35	1.3	0.012	126	82
SSE4-2.3-15	N/A	N/A	N/A	N/A

  - TCM: Total Cache Miss
  - TCA: Total Cache Access
  - Level 3 latency = 38 cycles
  - No L3 related PAPI counters on SEE4-2.3-15

Performance Comparison: TLB Miss

- TLB: translation look-aside buffer
  - cache for page tables which map addresses between virtual memory and physical memory
  - CPU → TLB → L1/2/3 cache → RAM → (page fault) → HDD
- TLB Miss : in 1M counters

Processor	GV (IM)	G4(IM)	GV (DM)	G4(DM)
AVX-2.0-15	53	4256	3168	4626
AVX2-2.4-35	0	0	44	91
SSE4-2.3-15	55	149	88	1628

  - IM/DM: Instruction/Data TLB Miss
- Cost for TLB Miss : (e.g. for AVX-2.0GHz-15MB
  - TLB_MISS_LATENCY_TIME = 2.85 (ns)
  - TLB_MISS_LATENCY_CYCLES = 6
    TLB_MISSES_COST_ONE_SECOND = 333.5 M counters

Performance Comparison: Remaining Issues

- Scaling problem
  - Scaling issues of GeantV, especially with the multithreaded vector-mode are now almost resolved
  - CPU Time [sec]: 1-Thread (1T) vs. 4-Threads (4T) with the vector mode (1101)

Processor	GV-vec 1T	GV-vec 4T	GV-vec 4T/1T
AVX-2.0-15	2331	2580	1.11
AVX2-2.4-35	1530	2302	1.50*
SSE4-2.3-15	4333	4394	1.01

  - *) AVX2 has only 2-cores
- Total memory usage (churn) in [MB]

Geant4	GeantV-scalar	GeantV-vector
280	882	2119*

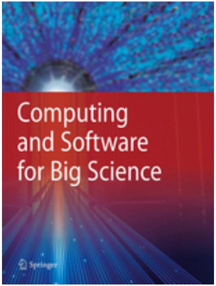
  - *) primary offender: NumaUtils::NumaAlignedMalloc (40%)
  - RollingIntegrationDriver<DormandPrince5RK> (20%)

Summary

- GeantV performance (EM physics with the CMS application): GeantV/Geant4 = 1.4 - 2.1
- The overall gain by vectorization is marginal
- The achieved performance gain is likely due to the relatively light structure of GeantV codes and the smaller size of libraries (caching effects)
- There may be still rooms to improve performance further. Nonetheless, the additional gain by explicit vectorization may be challenging due to data intensive and path-dependent stochastic nature of HEP detector simulation work flows.

Status of a general GeantV paper

- Motivation
  - a grand summary of the GeantV prototype and results
  - detailed descriptions of sub-projects or related works
  - an input document for the community meeting (Oct. 15, 2019)
- Target Journal(s)
  - Computing and Software for Big Science
  - arXiv (a detail technical note, if needed)
- Proposed time-line
  - the final draft by Oct. 1, 2019
  - feedback from the community meeting
  - submission the journal by Nov. 11, 2019



Sections of the Paper (Contributors): red → empty

- 1. Introduction (Andrei, Philippe, Witek)
    - 1.1 Motivation
- 2. Concepts and Architecture (Andrei, Philippe)
    - 2.1 Software design
        - 2.1.1 GeantV scheduler
        - 2.1.2 Scalar and vector workflows
        - 2.1.3 Concurrency mode
    - 2.2 Target hardware
- 3 Implementation
    - 3.1 Vector Libraries
        - 3.1.1 VecCore (Guilherme A.)
    - 3.2 Geometry Description : VecGeom (Sandro)
        - 3.2.1 Introduction (Sandro)
        - 3.2.2 Microbenchmarks on shape level
        - 3.2.3 SIMD navigation in basket mode
        - 3.2.4 SIMD navigation in single particle mode
        - 3.2.5 Specialization of code

Sections of the Paper (Contributors)

- 3 Implementation ... continue ...
  - 3.3 VecMath (Soon, Andrei)
    - 3.3.1 Fast Math
    - 3.3.2 Pseudo random number generation (Soon)
  - 3.4 GeantV tracking and navigation (Andrei)
  - 3.5 Physics Interface (Mihaly, Alberto R.)
  - 3.6 EM Physics models and vectorization (Mihaly, Marilena)
  - 3.7 Magnetic field integration (John)
  - 3.8 I/O (Witek, Philippe)
    - 3.8.1 Input
    - 3.8.2 Output
    - 3.8.3 MC truth
  - 3.9 User interface (Witek, Andrei)

Sections of the Paper (Contributors)

- 4. GeantV application and physics validations (Mihaly)
- 5. Usability aspects
  - 5.1 Reproducibility (Soon, John)
  - 5.2 Experiment framework integration (Sunanda, Kevin)
- 6. Performance results (Andrei, Soon, Guilherme L., Alberto M., Sunanda)
  - 6.1 Global performance
  - 6.2 Scheduler performance
  - 6.3 Profiling analysis
  - 6.4 Vectorization performance
  - 6.5 Concurrency performance
  - 6.6 Performance from user perspective
- 7. Lessons Learned
  - 7.1 Framework and work flows (Andrei, Philippe)
  - 7.2 Geometry and navigation (Sandro, Gabriele, Guilherme L)
  - 7.3 Vectorization of EM models (Marilena)
- 8. Summary and conclusion (Pere, Daniel, Witek)

Editorial Time-line and Status

- Proposed time-line (Not respected!)
  - The first draft for each section: by July 2
  - The first internal review: by July 30
  - The second draft: by August 27
  - The second review: by September 10
  - The final draft: by October 1
  - Feedback from the community meeting: on October 15
  - The last review: by October 29
  - Submit to the journal: by November 11, 2019
- The current draft on the overleaf:
  (https://www.overleaf.com/project/5cdefe7c9968db58bab664f2)