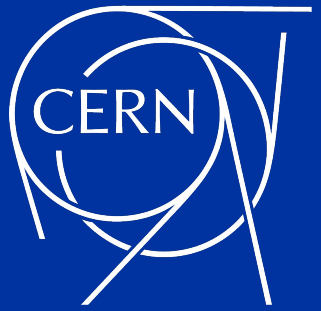


Geant4 10.7 Performance Report

G. Amadio (EP-SFT)

Jan, 2021

Hardware (1)

- Openlab machine (olhswep04.cern.ch)
- Dual socket Intel Xeon CPU E5-2698 v3 (Haswell, Q3 2014)
- 16 cores per socket, 2.3GHz base, 3.6GHz boost
- 64GB DRAM, 40MB L3, 4MB L2, 512K L1d/L1i
- Intel® SSD DC P3500 Series (NVMe)
- Hyperthreading disabled in BIOS
- This is the node in use unless indicated otherwise

Hardware (2)

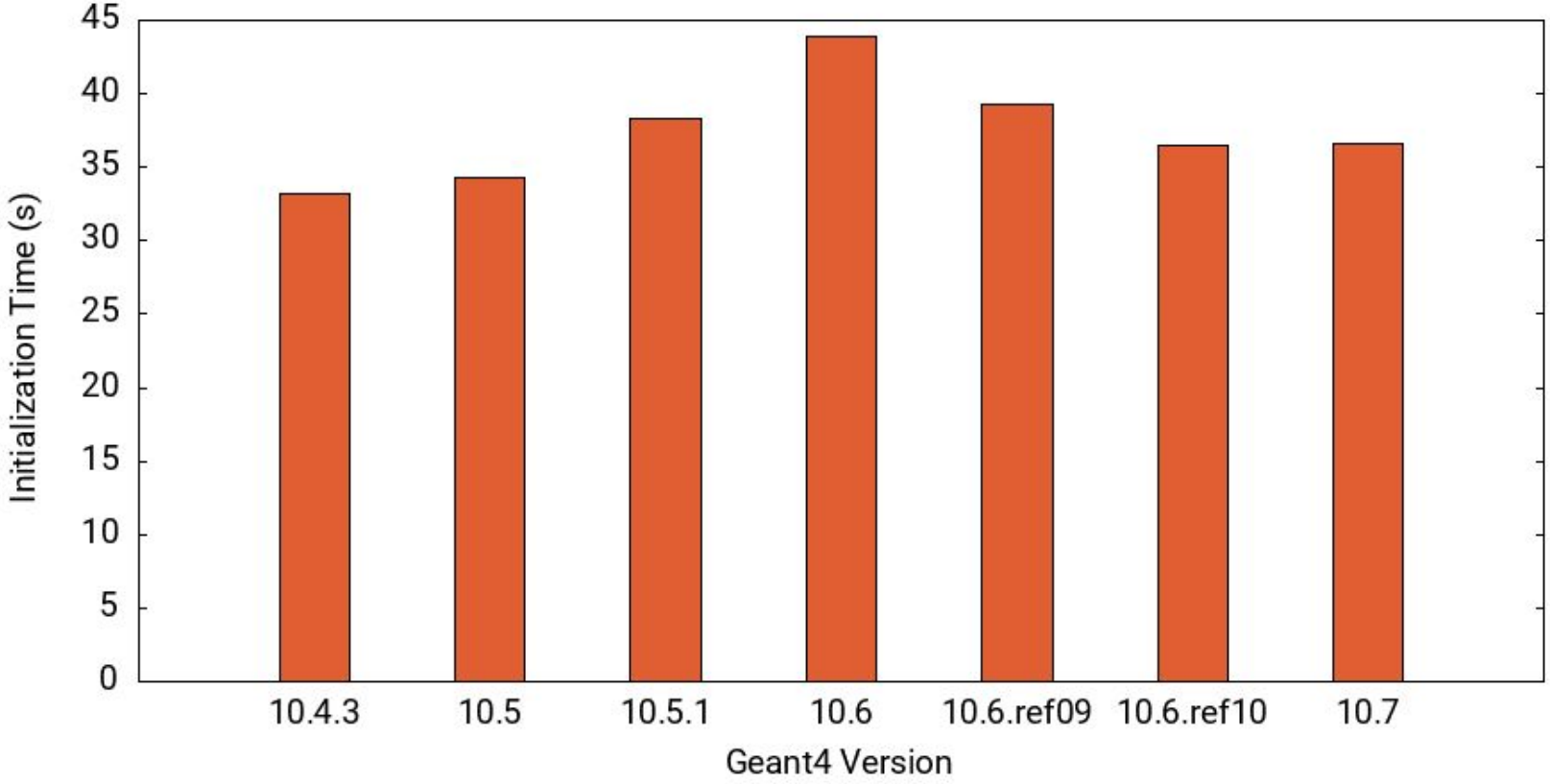
- Openlab machine (olsky-05.cern.ch)
- Dual socket Intel Xeon Silver 4110 (Skylake, Q3 2017)
- 8 cores per socket, 2.1GHz base, 3.0 GHz boost
- 64GB DRAM, 22MB L3, 16MB L2, 512K L1d/L1i
- Hyperthreading disabled in BIOS

Software

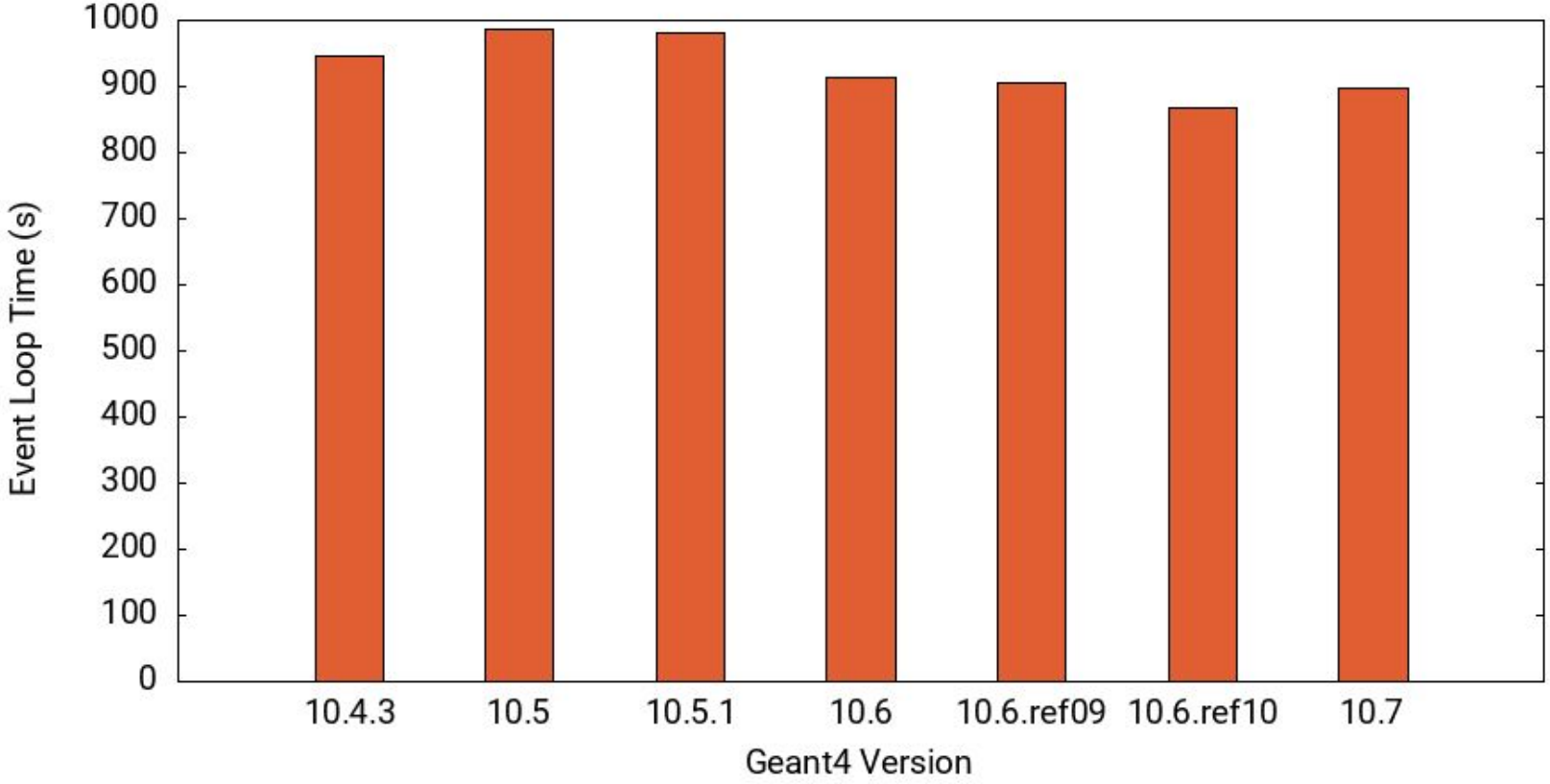
- CentOS 7.9.2009 / Linux 3.10.0-1160.6.1.el7.x86_64
- Toolchain and Geant4 dependencies from Gentoo Prefix
  - /cvmfs/sft.cern.ch/lcg/contrib/gentoo/linux/x86_64/startprefix
  - gcc-10.2.0, binutils-2.34, glibc-2.25 (limited by need to support kernel 2.6)
- Geant4 10.4 – 10.7 compiled locally, installed into SSD, data from CVMFS
  - Geant4 version is 10.7 unless indicated otherwise
  - CXXFLAGS: -O2 -std=c++11 -DNDEBUG -march=native -fno-omit-frame-pointer -g
- The benchmark
  - Simulation of 128 Pythia ttbar events at E = 14 TeV
  - CMS 2018 geometry, with B_z = 4 T (constant) and FTFP_BERT physics list
- Performance analysis with Intel VTune and perf

Performance Trends

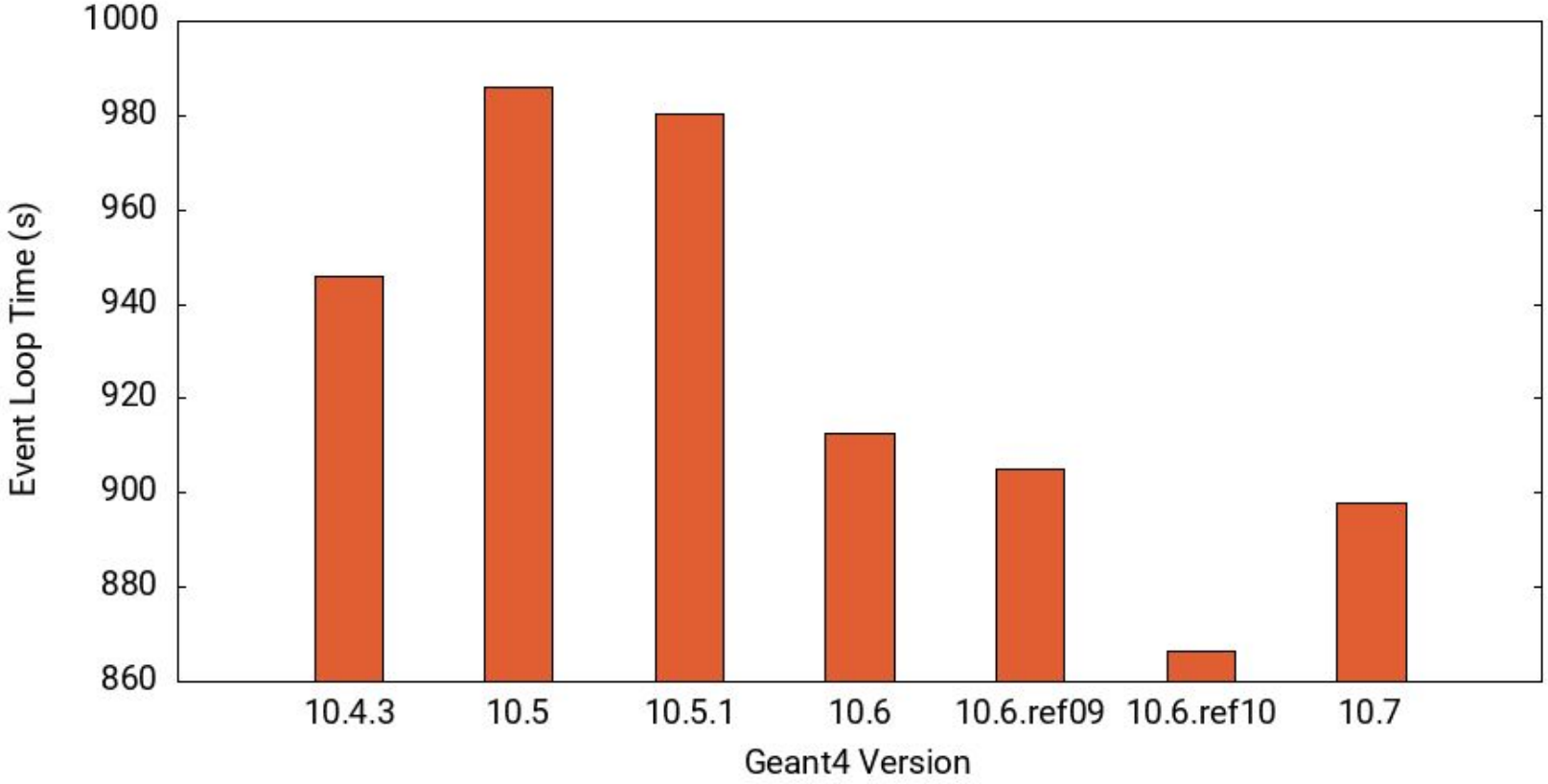
Geant4 Initialization Time



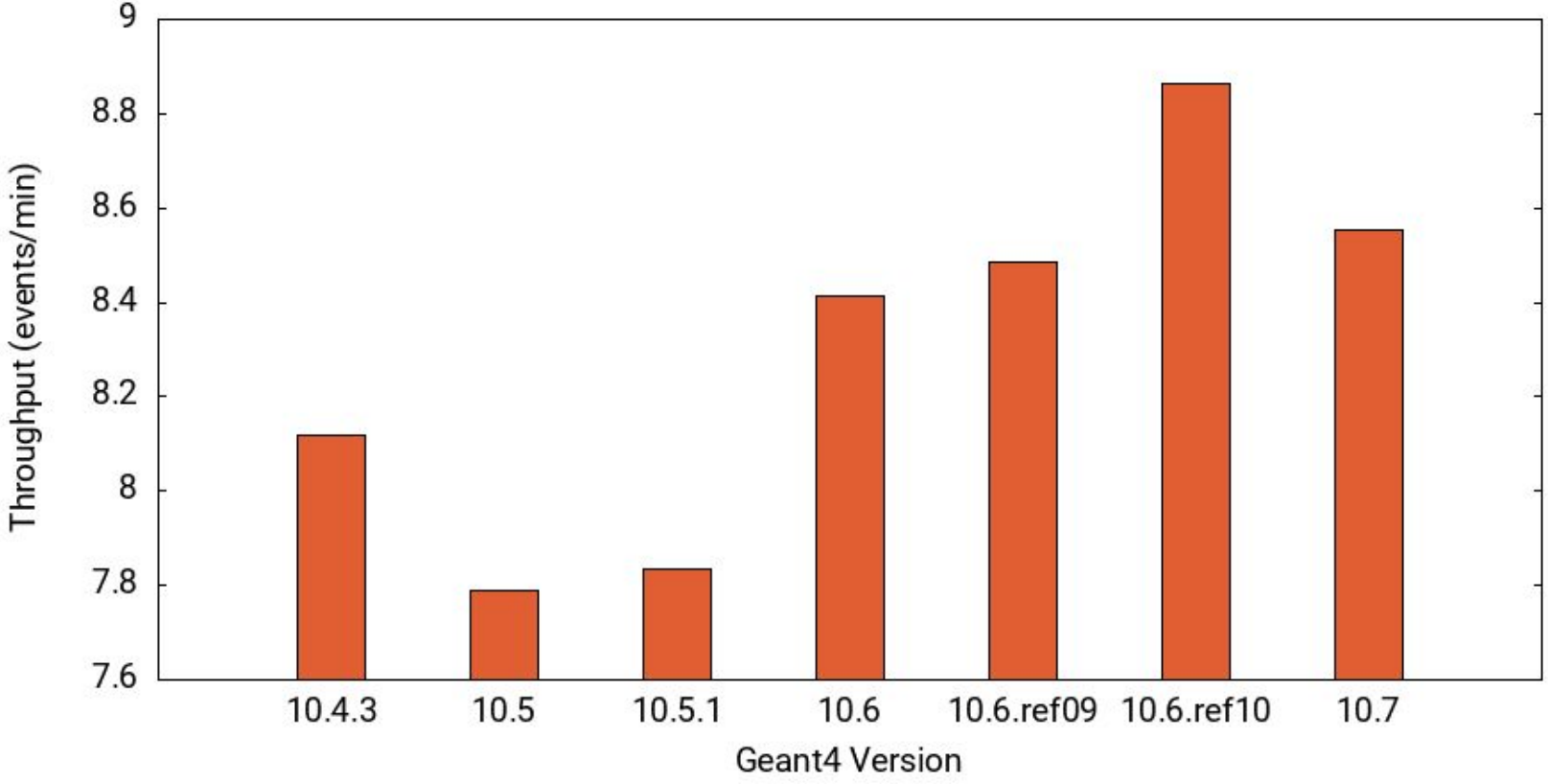
Geant4 Event Loop Time



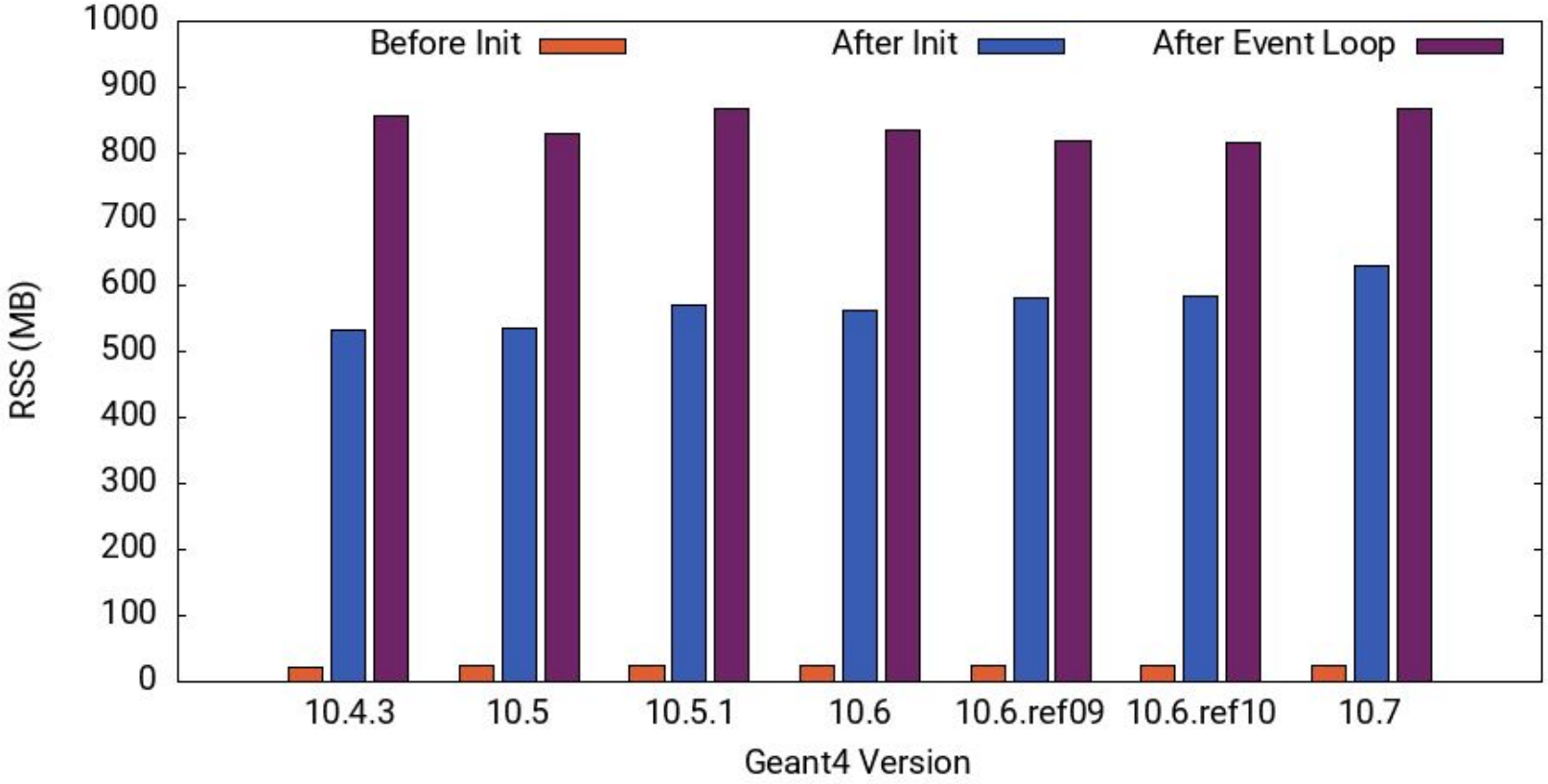
Geant4 Event Loop Time (reduced range)



Geant4 Throughput



Geant4 Memory Consumption

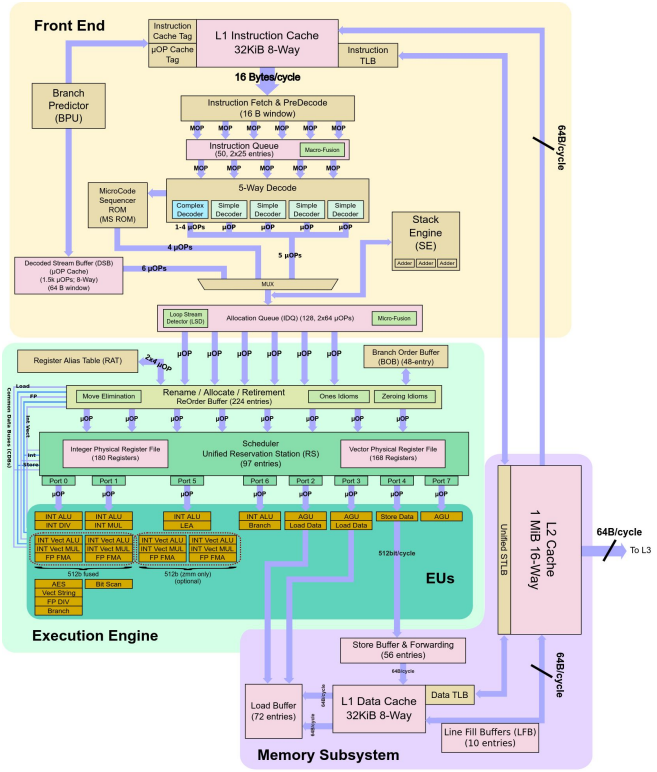


Top-down Microarchitecture Analysis

Intel and AMD Microarchitectures

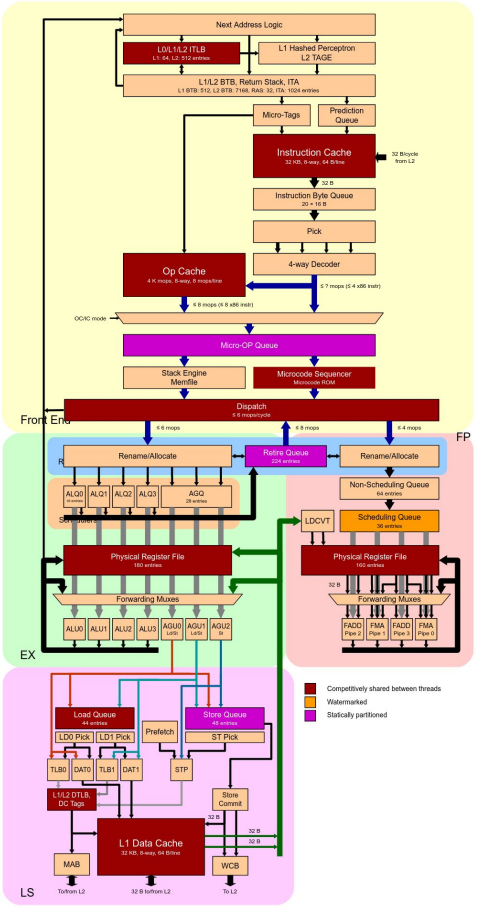
- Front End
  - Instruction Fetch and Decode
  - Branch Predictor Unit
  - L1 Instruction Cache
  - Instruction TLB
- Back End
  - Execution Engine
    - Register Renaming
    - Move Elimination
  - Memory Subsystem
    - Load/Store Units
    - L1 Data Cache
    - L2 Shared Cache
    - Data TLB

Intel Skylake



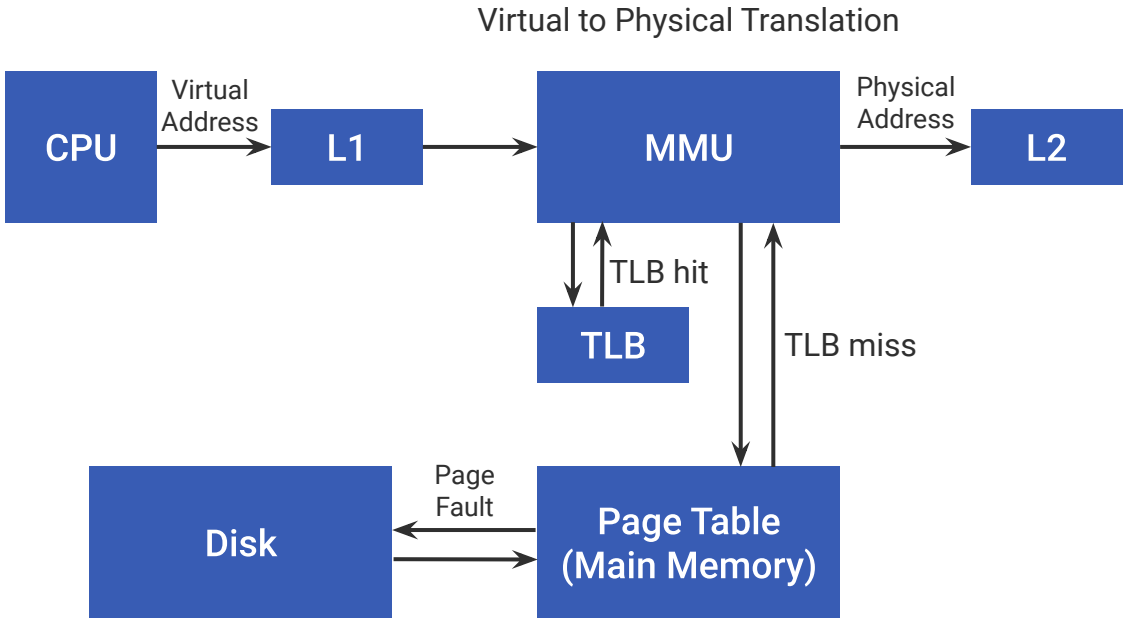
source: https://en.wikichip.org

AMD Zen 2



The Translation Lookaside Buffer

“A translation lookaside buffer (TLB) is a memory cache that is used to reduce the time taken to access a user memory location. It is a part of the chip's memory management unit (MMU). The TLB stores the recent translations of virtual memory to physical memory and can be called an address-translation cache.”

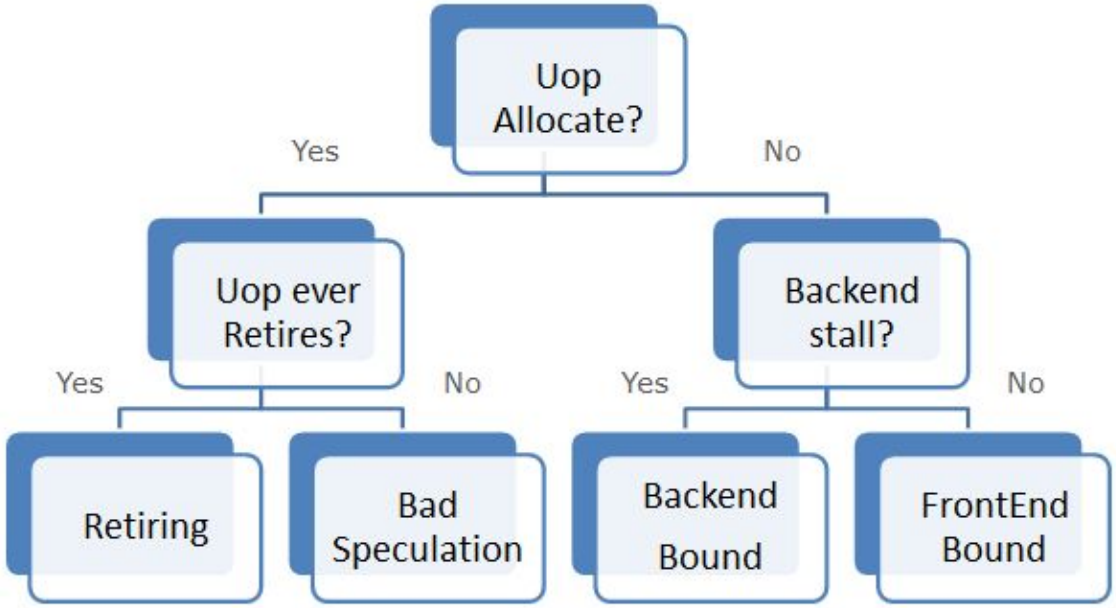


https://en.wikipedia.org/wiki/Translation_lookaside_buffer

Microarchitecture Analysis with perf

Metrics are only available with **perf stat**. To be able to get metrics per-symbol with **perf record**:

- Use classification from Intel VTune
- Use formulas for each category based on events known to perf and properties of the hardware
- Record all perf events in the same sampling group
- Report counts per symbol using perf
- Post-process with AWK to calculate metrics per symbol
- Can also use similar events and own formulas to create new relevant metrics

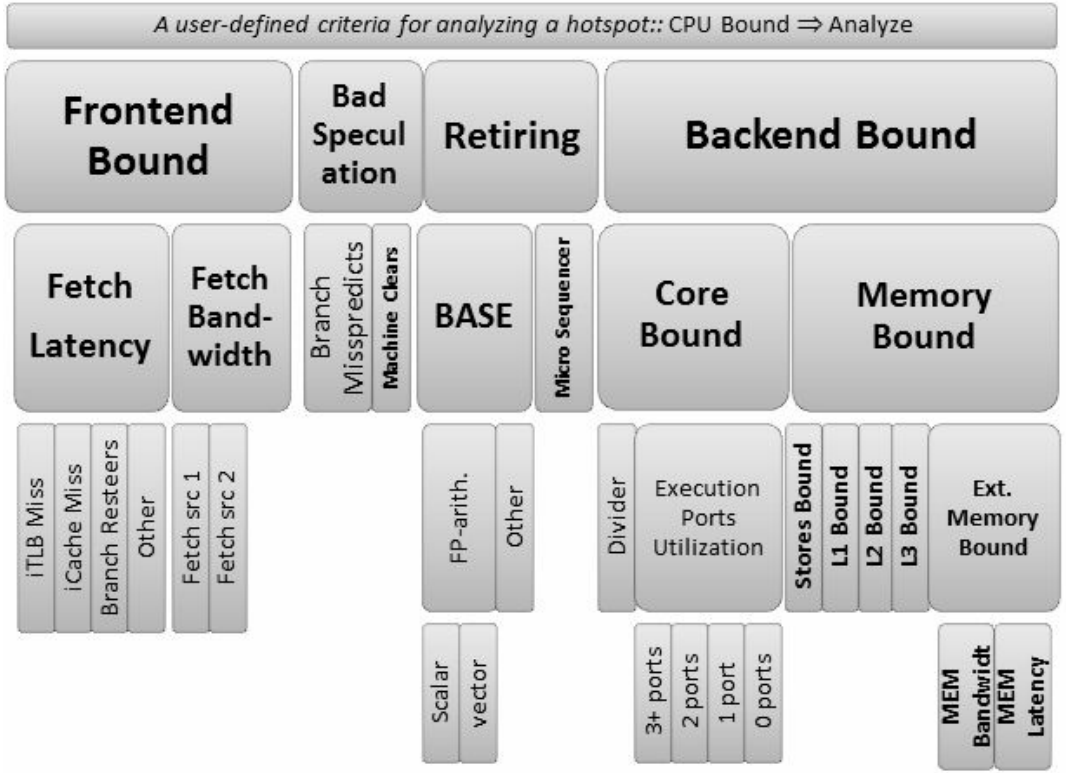


A. Yasin, "A Top-Down method for performance analysis and counters architecture," *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Monterey, CA, 2014, pp. 35-44, doi: 10.1109/ISPASS.2014.6844459.

Microarchitecture Analysis with perf

Metrics are only available with **perf stat**. To be able to get metrics per-symbol with **perf record**:

- Use classification from Intel VTune
- Use formulas for each category based on events known to perf and properties of the hardware
- Record all perf events in the same sampling group
- Report counts per symbol using perf
- Post-process with AWK to calculate metrics per symbol
- Can also use similar events and own formulas to create new relevant metrics



A. Yasin, "A Top-Down method for performance analysis and counters architecture," *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Monterey, CA, 2014, pp. 35-44, doi: 10.1109/ISPASS.2014.6844459.

Microarchitecture Analysis with perf

Metrics are only available with **perf stat**. To be able to get metrics per-symbol with **perf record**:

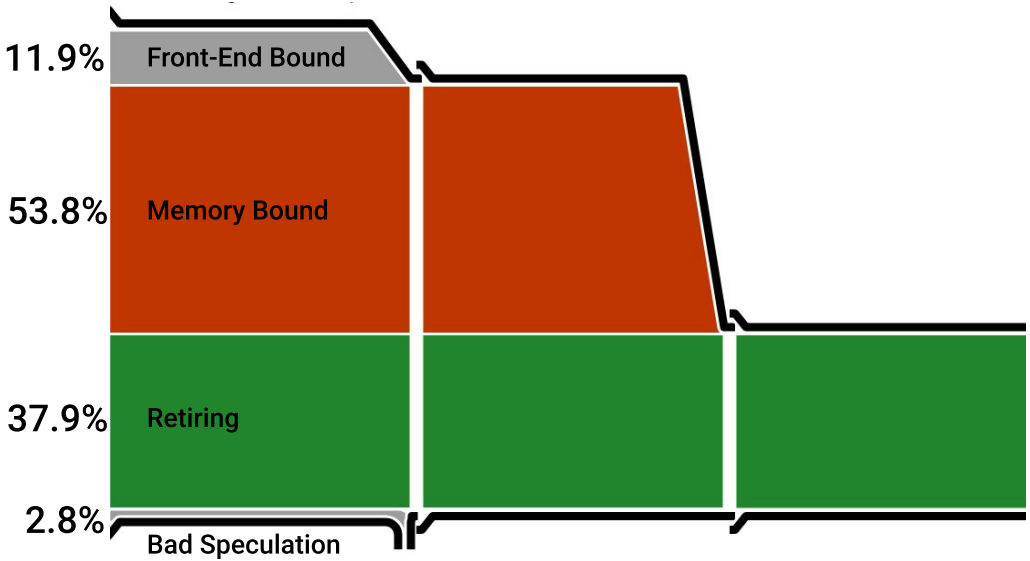
- Use classification from Intel VTune
- Use formulas for each category based on events known to perf and properties of the hardware
- Record all perf events in the same sampling group
- Report counts per symbol using perf
- Post-process with AWK to calculate metrics per symbol
- Can also use similar events and own formulas to create new relevant metrics

Table 7: Intel's implementation of Top-Down Metrics

Metric Name	Intel Core™ events
Clocks	*CPU_CLK_UNHALTED.THREAD*
Slots	*4 * Clocks*
Frontend Bound	IDQ_UOPS_NOT_DELIVERED.CORE / Slots
Bad Speculation	(UOPS_ISSUED.ANY - UOPS_RETIRED.RETIRE_SLOTS + 4* INT_MISC.RECOVERY_CYCLES) / Slots
Retiring	UOPS_RETIRED.RETIRE_SLOTS / Slots
Frontend Latency Bound	IDQ_UOPS_NOT_DELIVERED.CORE: [≥ 4] / Clocks
#BrMispredFraction	*BR_MISP_RETIRED.ALL_BRANCHES / (BR_MISP_RETIRED.ALL_BRANCHES + MACHINE_CLEARS.COUNT)*
MicroSequencer	#RetireUopFraction * IDQ.MS_UOPS / Slots
#ExecutionStalls	*(CYCLE_ACTIVITY.CYCLES_NO_EXECUTE - RS_EVENTS.EMPTY_CYCLES + UOPS_EXECUTED.THREAD: [≥ 1] - UOPS_EXECUTED.THREAD: [≥ 2]) / Clocks*
Memory Bound	*(CYCLE_ACTIVITY.STALLS_MEM_ANY + RESOURCE_STALLS.SB)* / Clocks
L1 Bound	(CYCLE_ACTIVITY.STALLS_MEM_ANY - CYCLE_ACTIVITY.STALLS_L1D_MISS) / Clocks
L2 Bound	(CYCLE_ACTIVITY.STALLS_L1D_MISS - CYCLE_ACTIVITY.STALLS_L2_MISS)/Clocks
#L3HitFraction	*MEM_LOAD_UOPS_RETIRED.LLC_HIT / (MEM_LOAD_UOPS_RETIRED.LLC_HIT +7*MEM_LOAD_UOPS_RETIRED.LLC_MISS)*
L3 Bound	(1 - #L3HitFraction) * CYCLE_ACTIVITY.STALLS_L2_MISS / Clocks
Ext. Memory Bound	CYCLE_ACTIVITY.STALLS_MEM_ANY
MEM Bandwidth	UNC_ARB_TRK_OCCUPANCY.ALL: [≥ 28]/ UNC_CLOCK.SOCKET
MEM Latency	(UNC_ARB_TRK_OCCUPANCY.ALL: [≥ 1] - UNC_ARB_TRK_OCCUPANCY.ALL: [≥ 28]) / UNC_CLOCK.SOCKET

A. Yasin, "A Top-Down method for performance analysis and counters architecture," *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Monterey, CA, 2014, pp. 35-44, doi: 10.1109/ISPASS.2014.6844459.

Microarchitecture Usage on Haswell

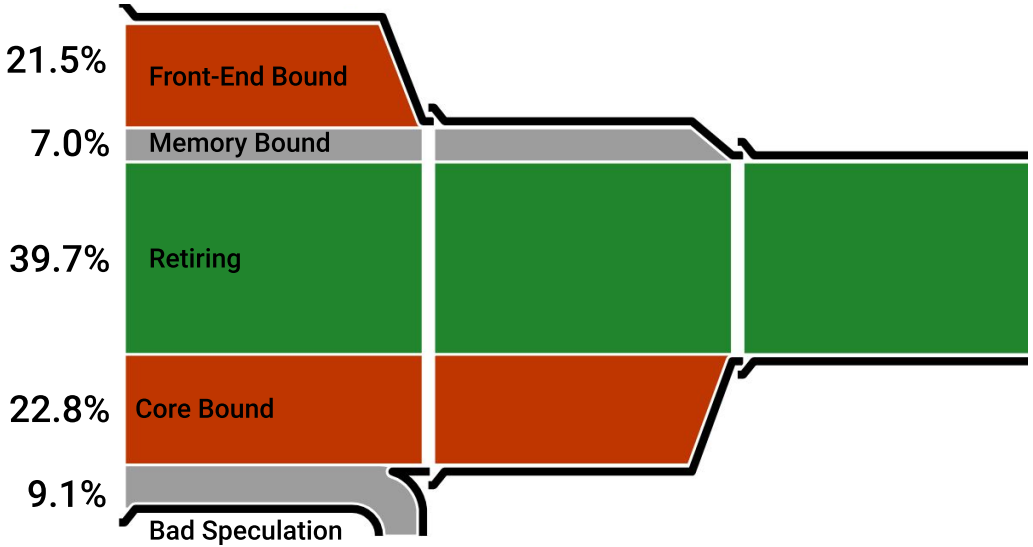


Retiring:	**37.9%**	**of Pipeline Slots**
Front-End Bound:	**11.9%**	**of Pipeline Slots**
Bad Speculation:	**2.8%**	**of Pipeline Slots**
Back-End Bound:	**47.3%**	**of Pipeline Slots**
Memory Bound:	**53.8%**	**of Pipeline Slots**
L1 Bound:	**48.9%**	**of Clockticks**
DTLB Overhead:	13.3%	of Clockticks
Loads Blocked by Store Forwarding:	0.0%	of Clockticks
Lock Latency:	0.0%	of Clockticks
Split Loads:	0.0%	of Clockticks
4K Aliasing:	0.3%	of Clockticks
FB Full:	0.0%	of Clockticks
L2 Bound:	0.0%	of Clockticks
L3 Bound:	**5.6%**	**of Clockticks**
Contested Accesses:	0.0%	of Clockticks
Data Sharing:	0.0%	of Clockticks
L3 Latency:	6.9%	of Clockticks
SQ Full:	0.0%	of Clockticks
DRAM Bound:	**0.0%**	**of Clockticks**
Store Bound:	**0.0%**	**of Clockticks**
Core Bound:	**0.0%**	**of Pipeline Slots**

Mostly memory bound on Haswell

https://software.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/methodologies/top-down-microarchitecture-analysis-method.html

Microarchitecture Usage on Skylake



Retiring:	**39.7%**	**of Pipeline Slots**
Front-End Bound:	**21.5%**	**of Pipeline Slots**
Bad Speculation:	**9.1%**	**of Pipeline Slots**
Back-End Bound:	**29.8%**	**of Pipeline Slots**
Memory Bound:	**7.0%**	**of Pipeline Slots**
Core Bound:	**22.8%**	**of Pipeline Slots**
Divider:	10.3%	of Clockticks
Port Utilization:	**24.5%**	**of Clockticks**
Cycles of 0 Ports Utilized:	**1.9%**	**of Clockticks**
Cycles of 1 Port Utilized:	17.1%	of Clockticks
Cycles of 2 Ports Utilized:	18.7%	of Clockticks
Cycles of 3+ Ports Utilized:	**38.5%**	**of Clockticks**
Vector Capacity Usage (FPU):	12.5%	

Mostly frontend and core bound on Skylake, quite different than Haswell

https://software.intel.com/content/www/us/en/develop/documentation/vtune-cookbook/top/methodologies/top-down-microarchitecture-analysis-method.html

Top 20 functions measured with VTune on Haswell

Source Function	CPU Time ▼	Instructions Retired	CPI Rate	Retiring	Front-End Bound	Bad Speculation	Back-End Bound	
							Memory Bound	Core Bound
__ieee754_atan2_avx	4.9%	5.1%	0.713	0.379	11.9%	2.8%	53.8%	0.0%
G4CrossSectionDataStore::GetCrossSection	2.7%	3.5%	0.568	0.494	13.5%	8.7%	52.0%	0.0%
G4CrossSectionDataStore::GetIsoCrossSection	2.2%	3.7%	0.435	0.536	15.2%	2.7%	26.5%	2.0%
G4Log	1.9%	1.2%	1.221	0.232	26.4%	12.7%	49.3%	0.0%
std::min<double>	1.8%	1.8%	0.741	0.402	7.7%	7.9%	68.4%	0.0%
G4PhysicsVector::Interpolation	1.7%	1.1%	1.134	0.225	8.4%	17.3%	62.9%	0.0%
G4PolyhedraSide::DistanceAway	1.5%	1.5%	0.741	0.335	4.3%	34.1%	84.2%	0.0%
G4PolyPhiFace::InsideEdges	1.5%	2.1%	0.511	0.485	18.0%	7.9%	23.3%	2.2%
std::max<double>	1.4%	1.1%	0.952	0.305	5.8%	9.0%	73.6%	0.0%
G4Exp	1.3%	0.9%	1.158	0.263	13.2%	13.5%	54.5%	0.0%
CLHEP::operator-	1.3%	1.5%	0.626	0.400	6.3%	0.0%	66.1%	0.0%
CLHEP::Hep3Vector::operator[]	1.2%	1.3%	0.667	0.379	22.3%	9.8%	51.7%	0.0%
G4SteppingManager::DefinePhysicalStepLength	1.1%	0.9%	0.887	0.291	23.3%	14.9%	46.3%	0.0%
CLHEP::Hep3Vector::dot	0.9%	0.8%	0.884	0.437	10.3%	0.0%	65.1%	0.0%
G4CrossSectionDataStore::ComputeCrossSection	0.8%	1.0%	0.579	0.434	26.7%	3.3%	42.2%	0.0%
G4VCSGfaceted::Inside	0.8%	0.5%	1.042	0.244	11.9%	41.1%	39.4%	0.0%
G4PolyhedraSide::PhiSegment	0.8%	1.6%	0.352	0.668	7.0%	0.0%	39.9%	9.1%
CLHEP::MixMaxRng::iterate	0.8%	1.1%	0.492	0.421	49.3%	10.5%	0.0%	0.0%
G4PolyhedraSide::Inside	0.8%	0.7%	0.801	0.319	8.4%	27.8%	66.1%	0.0%
G4VEmProcess::PostStepGetPhysicalInteractionLength	0.7%	0.5%	1.212	0.191	15.5%	14.6%	62.9%	0.0%

Top 20 functions measured with VTune on Skylake

Source Function	CPU Time ▼	Instructions Retired	CPI Rate	Retiring	Front-End Bound	Bad Speculation	Back-End Bound	
							Memory Bound	Core Bound
__ieee754_atan2_avx	4.7%	4.8%	0.651	0.406	11.2%	4.6%	8.6%	35.0%
G4CrossSectionDataStore::GetCrossSection	2.9%	3.8%	0.500	0.557	12.0%	9.3%	4.3%	18.8%
G4CrossSectionDataStore::GetIsoCrossSection	2.4%	3.9%	0.418	0.567	11.3%	4.7%	4.1%	23.1%
G4PolyPhiFace::InsideEdges	1.9%	1.9%	0.680	0.370	10.3%	10.0%	19.8%	22.9%
G4Log	1.8%	1.1%	1.062	0.261	24.5%	24.0%	4.4%	21.0%
G4PhysicsVector::Interpolation	1.8%	1.1%	1.058	0.270	9.2%	11.4%	11.5%	40.9%
CLHEP::operator-	1.6%	1.4%	0.771	0.468	7.9%	0.0%	15.6%	33.3%
std::min<double>	1.5%	1.1%	0.883	0.343	8.2%	8.1%	11.7%	37.7%
std::max<double>	1.5%	1.3%	0.760	0.371	7.5%	11.0%	10.6%	33.8%
G4PolyhedraSide::DistanceAway	1.4%	1.5%	0.662	0.389	6.7%	35.4%	3.6%	15.5%
CLHEP::Hep3Vector::operator[]	1.3%	1.5%	0.581	0.456	17.4%	7.8%	6.6%	22.5%
G4Exp	1.2%	0.9%	0.906	0.315	15.5%	15.0%	4.6%	33.4%
G4SteppingManager::DefinePhysicalStepLength	1.1%	1.1%	0.650	0.392	21.6%	17.1%	3.8%	18.3%
G4VCSGfaceted::Inside	1.0%	0.6%	1.126	0.229	10.1%	24.9%	21.5%	20.5%
G4PhysicsVector::ComputeLogVectorBin	1.0%	1.0%	0.691	0.418	10.0%	5.8%	10.8%	31.5%
G4AffineTransform::TransformPoint	0.9%	0.8%	0.771	0.361	12.9%	12.8%	11.6%	26.6%
CLHEP::MixMaxRng::iterate	0.9%	1.1%	0.513	0.658	42.4%	0.0%	0.2%	3.1%
CLHEP::Hep3Vector::dot	0.8%	0.7%	0.790	0.265	8.3%	28.9%	13.1%	23.2%
G4VEmProcess::PostStepGetPhysicalInteractionLength	0.8%	0.5%	1.098	0.230	18.2%	15.4%	13.9%	29.5%
G4PolyhedraSide::Inside	0.8%	0.6%	0.789	0.410	8.8%	24.8%	4.2%	21.2%

Differences between Geant4 10.6.ref10 and 10.7

Elapsed Time: 951.254s - 938.336s = 12.918s

CPU Time:	**27478.941s - 27112.274s = 366.667s**
Total Thread Count:	Not changed, 182
Paused Time:	Not changed, 0s

Difference is (10.6.re10 - 10.7)
10.7 was faster in VTune for this run

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
__ieee754_atan2_avx	libm-2.25.so	1318.204s - 1328.819s = -10.615s
G4CrossSectionDataStore::GetCrossSection	libG4processes.so	726.167s - 733.714s = -7.547s
G4CrossSectionDataStore::GetIsoCrossSection	libG4processes.so	601.880s - 588.710s = 13.170s
G4PolyhedraSide::DistanceAway	libG4geometry.so	405.687s - 409.636s = -3.949s
G4PolyPhiFace::InsideEdges	libG4geometry.so	394.732s - 403.532s = -8.800s
[Others]	N/A*	24032.271s - 23647.863s = 384.408s

*N/A is applied to non-summable metrics.

Top Hotspots by Difference

This section displays the performance difference between two selected results for the most active functions in your application.

Function	Module	CPU Time, sorted by abs. difference
std::min<double>	libG4physicslists.so	295.713s - 1.874s = 293.839s
std::min<double>	libG4physicslists.so	54.195s - 311.139s = -256.943s
std::max<double>	libG4physicslists.so	265.583s - 19.545s = 246.038s
std::max<double>	libG4physicslists.so	28.766s - 263.970s = -235.203s
G4UniversalFluctuation::SampleFluctuations	libG4processes.so	10.264s - 167.647s = -157.383s
[Others]	N/A*	26824.419s - 26348.099s = 476.320s

*N/A is applied to non-summable metrics.

Improvements in Geant4 10.7 vs 10.6.ref10

Class	CPU Time: Difference ▼ Effective Time	CPU Time: 10.6.ref10	CPU Time: 10.7	Instructions Retired: 10.6.ref10	Instructions Retired: 10.7	Instructions Retired: Difference	CPI Rate: 10.6.ref10	CPI Rate: 10.7	CPI Rate: Difference
CLHEP::Hep3Vector	200.764s	1542.973s	1342.210s	5,749,034,000,000	5,385,036,000,000	363,998,000,000	0.715	0.670	0.045
G4Navigator	146.278s	575.489s	429.211s	1,934,116,000,000	1,771,667,000,000	162,449,000,000	0.795	0.647	0.148
G4ParticleChangeForTransport	42.749s	135.543s	92.794s	376,970,000,000	267,030,000,000	109,940,000,000	0.949	0.930	0.019
G4AffineTransform	37.867s	468.783s	430.915s	1,726,932,000,000	1,707,060,000,000	19,872,000,000	0.732	0.676	0.056
G4AllocatorPool	33.788s	119.145s	85.357s	292,169,000,000	240,787,000,000	51,382,000,000	1.089	0.944	0.145
G4FieldManagerStore	30.661s	73.049s	42.388s	148,833,000,000	134,780,000,000	14,053,000,000	1.337	0.860	0.476
G4Transportation	30.120s	377.382s	347.262s	1,454,819,000,000	1,457,510,000,000	-2,691,000,000	0.690	0.640	0.051
G4Track	25.900s	180.617s	154.717s	679,006,000,000	609,132,000,000	69,874,000,000	0.685	0.675	0.009
G4PropagatorInField	17.029s	67.696s	50.667s	216,982,000,000	194,856,000,000	22,126,000,000	0.847	0.711	0.135
CLHEP::MixMaxRng	15.275s	451.553s	436.278s	2,118,116,000,000	2,081,638,000,000	36,478,000,000	0.566	0.567	-0.001
G4PhysicsVector	15.035s	795.517s	780.482s	2,125,384,000,000	2,120,209,000,000	5,175,000,000	0.999	0.983	0.016
G4VCrossSectionDataSet	14.594s	48.271s	33.678s	222,456,000,000	116,403,000,000	106,053,000,000	0.562	0.749	-0.187
G4StepPoint	14.293s	180.547s	166.254s	862,845,000,000	836,533,000,000	26,312,000,000	0.564	0.524	0.040
G4LogicalVolume	13.281s	366.557s	353.276s	891,710,000,000	1,087,693,000,000	-195,983,000,000	1.097	0.874	0.224
G4VEmProcess	12.268s	439.996s	427.728s	1,056,620,000,000	1,022,120,000,000	34,500,000,000	1.112	1.121	-0.008
G4SteppingManager	12.118s	866.822s	854.704s	3,250,038,000,000	3,495,241,000,000	-245,203,000,000	0.714	0.652	0.061
std::vector<std::vector<G4NavigationLevel, std::allocator<G4	9.682s	20.377s	10.695s	72,496,000,000	41,653,000,000	30,843,000,000	0.725	0.688	0.037
G4KleinNishinaCompton	9.452s	73.670s	64.218s	107,571,000,000	106,421,000,000	1,150,000,000	1.791	1.594	0.197
G4NavigationHistoryPool	9.432s	24.587s	15.155s	80,454,000,000	57,569,000,000	22,885,000,000	0.783	0.740	0.042
G4FieldManager	9.392s	36.304s	26.912s	55,062,000,000	63,664,000,000	-8,602,000,000	1.736	1.112	0.624
G4NavigationHistory	7.357s	96.693s	89.336s	273,171,000,000	317,630,000,000	-44,459,000,000	0.930	0.764	0.166
CLHEP::RandGaussQ	7.327s	83.573s	76.246s	126,868,000,000	132,480,000,000	-5,612,000,000	1.791	1.590	0.201
G4ProductionCutsTable	7.106s	12.459s	5.352s	31,579,000,000	11,086,000,000	20,493,000,000	1.084	1.342	-0.259
G4PVPlacement	6.986s	46.447s	39.461s	57,477,000,000	66,240,000,000	-8,763,000,000	2.140	1.575	0.565
G4TouchableHistory	6.856s	256.071s	249.216s	863,489,000,000	830,622,000,000	32,867,000,000	0.796	0.804	-0.008
G4VParticleChange	6.665s	68.959s	62.294s	230,230,000,000	215,073,000,000	15,157,000,000	0.817	0.784	0.033
G4PolyconeSide	6.004s	683.498s	677.495s	3,239,435,000,000	3,224,278,000,000	15,157,000,000	0.564	0.569	-0.005
G4Region	5.803s	111.167s	105.363s	165,715,000,000	205,022,000,000	-39,307,000,000	1.794	1.374	0.420

Regressions in Geant4 10.7 vs 10.6.ref10

Class	CPU Time: Difference Effective Time	CPU Time: 10.6.ref10	CPU Time: 10.7	Instructions Retired: 10.6.ref10	Instructions Retired: 10.7	Instructions Retired: Difference	CPI Rate: 10.6.ref10	CPI Rate: 10.7	CPI Rate: Difference
G4VoxelNavigation	-55.518s	416.281s	471.800s	744,510,000,000	885,040,000,000	-140,530,000,000	1.485	1.413	0.071
G4PolyhedraSide	-30.380s	1408.122s	1438.502s	6,395,242,000,000	6,534,300,000,000	-139,058,000,000	0.587	0.586	0.002
std	-24.547s	1033.336s	1057.883s	3,247,876,000,000	3,389,119,000,000	-141,243,000,000	0.862	0.831	0.030
G4ProcessManager	-21.620s	218.685s	240.305s	898,357,000,000	967,426,000,000	-69,069,000,000	0.654	0.662	-0.008
G4VEnergyLossProcess	-17.751s	315.920s	333.671s	920,276,000,000	945,576,000,000	-25,300,000,000	0.921	0.949	-0.027
G4NeutronInelasticXS	-16.949s	109.082s	126.031s	660,146,000,000	702,351,000,000	-42,205,000,000	0.449	0.491	-0.042
G4DynamicParticle	-15.506s	493.660s	509.166s	1,920,937,000,000	2,050,979,000,000	-130,042,000,000	0.693	0.659	0.034
G4NeutronCaptureXS	-12.810s	99.780s	112.590s	594,527,000,000	614,399,000,000	-19,872,000,000	0.457	0.506	-0.049
G4UniversalFluctuation	-12.479s	203.720s	216.199s	334,926,000,000	342,401,000,000	-7,475,000,000	1.631	1.666	-0.034
G4VPhysicalVolume	-11.336s	125.249s	136.585s	293,572,000,000	315,169,000,000	-21,597,000,000	1.149	1.156	-0.007
G4Proton	-11.316s	54.376s	65.692s	346,886,000,000	403,880,000,000	-56,994,000,000	0.423	0.431	-0.008
G4ElementData	-10.785s	60.420s	71.204s	530,518,000,000	585,557,000,000	-55,039,000,000	0.311	0.320	-0.009
G4CrossSectionDataStore	-8.530s	1547.774s	1556.304s	8,034,176,000,000	8,149,222,000,000	-115,046,000,000	0.515	0.512	0.003
G4VEmModel	-8.049s	118.644s	126.693s	368,920,000,000	367,540,000,000	1,380,000,000	0.856	0.923	-0.067
G4UrbanMscModel	-7.598s	344.606s	352.203s	657,202,000,000	665,896,000,000	-8,694,000,000	1.402	1.406	-0.004
G4NormalNavigation	-6.966s	172.178s	179.144s	560,694,000,000	607,384,000,000	-46,690,000,000	0.810	0.785	0.025
G4FieldTrack	-6.946s	33.387s	40.333s	126,500,000,000	163,921,000,000	-37,421,000,000	0.674	0.641	0.033
G4CascadeRecoilMaker	-6.244s	21.700s	27.945s	71,093,000,000	78,407,000,000	-7,314,000,000	0.839	0.965	-0.126
G4VMultipleScattering	-6.234s	86.249s	92.484s	227,631,000,000	278,185,000,000	-50,554,000,000	1.013	0.903	0.110
G4CollisionOutput	-5.453s	50.667s	56.120s	180,297,000,000	179,170,000,000	1,127,000,000	0.747	0.831	-0.084
G4Tubs	-5.312s	179.645s	184.957s	370,024,000,000	368,621,000,000	1,403,000,000	1.293	1.329	-0.036
G4CascadeFinalStateAlgorithm	-5.312s	16.448s	21.760s	63,365,000,000	68,080,000,000	-4,715,000,000	0.688	0.810	-0.122
G4MultiBodyMomentumDist	-5.242s	9.382s	14.624s	24,587,000,000	27,071,000,000	-2,484,000,000	0.949	1.397	-0.447
std::vector<G4SmartVoxelProxy*, std::allocator<G4SmartVoxe	-5.222s	44.112s	49.334s	75,256,000,000	80,638,000,000	-5,382,000,000	1.627	1.683	-0.057
G4InuclParticle	-5.132s	81.218s	86.349s	260,130,000,000	257,922,000,000	2,208,000,000	0.829	0.890	-0.061
G4NavigationLevelRep	-4.982s	236.737s	241.718s	877,841,000,000	977,408,000,000	-99,567,000,000	0.726	0.662	0.064
G4VCSGfaceted	-4.901s	382.032s	386.934s	1,094,294,000,000	1,109,658,000,000	-15,364,000,000	0.923	0.929	-0.006
G4PolyPhiFace	-4.881s	546.562s	551.444s	2,658,639,000,000	2,733,757,000,000	-75,118,000,000	0.552	0.536	0.017
std::vector<double, std::allocator<double>>	-4.771s	230.051s	234.822s	659,939,000,000	669,645,000,000	-9,706,000,000	0.922	0.930	-0.009

Regressions sorted by increase in retired instructions

Class	CPU Time: Difference Effective Time	CPU Time: 10.6.ref10	CPU Time: 10.7	Instructions Retired: 10.6.ref10	Instructions Retired: 10.7	Instructions Retired: Difference ▲	CPI Rate: 10.6.ref10	CPI Rate: 10.7	CPI Rate: Difference
G4PolyhedraSide	-42.248s	2103.839s	2146.086s	9,103,814,000,000	9,335,148,000,000	-231,334,000,000	0.616	0.614	0.002
[Not part of any known object class]	12.439s	3314.324s	3301.875s	11,424,146,000,000	11,649,960,000,000	-225,814,000,000	0.778	0.757	0.020
G4LogicalVolume	10.254s	312.542s	302.288s	798,514,000,000	982,790,000,000	-184,276,000,000	1.048	0.827	0.221
G4Step	16.638s	237.358s	220.720s	1,108,209,000,000	1,261,734,000,000	-153,525,000,000	0.574	0.464	0.109
G4SteppingManager	63.336s	1468.942s	1405.606s	5,915,669,000,000	6,067,469,000,000	-151,800,000,000	0.664	0.619	0.045
G4CrossSectionDataStore	-7.016s	1660.054s	1667.070s	8,626,012,000,000	8,742,277,000,000	-116,265,000,000	0.514	0.513	0.002
G4VMultipleScattering	0.862s	154.958s	154.096s	430,445,000,000	506,414,000,000	-75,969,000,000	0.964	0.831	0.132
G4PolyPhiFace	-5.693s	602.672s	608.365s	2,939,400,000,000	3,007,917,000,000	-68,517,000,000	0.552	0.535	0.017
G4ElectroNuclearCrossSection	-6.856s	162.014s	168.870s	546,250,000,000	614,330,000,000	-68,080,000,000	0.798	0.736	0.062
G4NeutronCaptureXS	-19.004s	186.200s	205.204s	1,187,559,000,000	1,254,466,000,000	-66,907,000,000	0.424	0.445	-0.021
G4Proton	-11.316s	54.376s	65.692s	346,886,000,000	403,880,000,000	-56,994,000,000	0.423	0.431	-0.008
G4ProcessManager	-25.850s	261.975s	287.825s	1,068,327,000,000	1,123,711,000,000	-55,384,000,000	0.658	0.683	-0.025
G4FieldTrack	11.727s	38.980s	27.253s	101,752,000,000	156,492,000,000	-54,740,000,000	1.012	0.470	0.541
G4PhysicsVector	2.646s	1609.718s	1607.071s	4,446,889,000,000	4,501,583,000,000	-54,694,000,000	0.970	0.954	0.017
G4NavigationLevel	0.311s	377.482s	377.171s	1,480,188,000,000	1,529,983,000,000	-49,795,000,000	0.680	0.659	0.022
G4NeutronInelasticXS	-17.500s	129.940s	147.441s	845,756,000,000	891,020,000,000	-45,264,000,000	0.420	0.451	-0.031
G4Box	-5.362s	240.435s	245.798s	1,027,272,000,000	1,068,649,000,000	-41,377,000,000	0.625	0.614	0.011
G4VEnergyLossProcess	-35.602s	456.534s	492.137s	1,382,737,000,000	1,422,481,000,000	-39,744,000,000	0.885	0.926	-0.041
G4Region	5.904s	112.500s	106.596s	171,695,000,000	209,944,000,000	-38,249,000,000	1.746	1.350	0.396
G4DynamicParticle	-5.132s	193.096s	198.228s	704,559,000,000	742,739,000,000	-38,180,000,000	0.740	0.701	0.039
G4PhotoNuclearCrossSection	-4.420s	153.043s	157.464s	830,254,000,000	865,904,000,000	-35,650,000,000	0.493	0.490	0.004
G4StackManager	5.102s	74.873s	69.771s	183,563,000,000	217,764,000,000	-34,201,000,000	1.103	0.844	0.259
G4NormalNavigation	-0.020s	203.069s	203.089s	795,731,000,000	826,436,000,000	-30,705,000,000	0.673	0.660	0.013
G4HadronElasticProcess	1.373s	42.729s	41.356s	109,963,000,000	138,184,000,000	-28,221,000,000	1.029	0.773	0.256
G4FieldManager	3.268s	22.923s	19.655s	30,797,000,000	57,293,000,000	-26,496,000,000	1.960	0.882	1.078
G4VDiscreteProcess	0.952s	240.776s	239.824s	324,829,000,000	349,508,000,000	-24,679,000,000	1.965	1.858	0.107
G4ParticleDefinition	-4.651s	30.460s	35.111s	80,477,000,000	103,891,000,000	-23,414,000,000	1.042	0.915	0.127

Unused cache in G4CrossSectionDS

- Cross section calculated for a given material, particle and element combination each time
- Loops over elements and/or isotopes and computes cross section for each one
- Current element or isotope therefore always changes between calls and the cache is never hit
- Also abundance in material description should be always larger than zero
- Fixed in master branch already

Source	Clockticks	Instructions Retired	CPI Rate	Locators			
				Retiring	Front-End Bound	Bad Speculation	Back-End Bound
G4double							
G4CrossSectionDataStore::GetCrossSection(const G4DynamicParticle* part,							
const G4Element* elm,							
const G4Material* mat)							
{	0.2%	0.3%	0.440	4.1%	0.9%	0.4%	1.8%
if(mat == elmMaterial && elm == currentElement &&	0.0%	0.0%	0.577	0.0%	0.0%	0.0%	0.0%
part->GetDefinition() == elmParticle &&	0.0%	0.1%	0.568	0.9%	0.2%	0.0%	0.6%
part->GetKineticEnergy() == elmKinEnergy)							
{ return elmCrossSection; }							
elmMaterial = mat;	0.0%	0.0%	0.298	0.1%	0.0%	0.0%	0.0%
currentElement = elm;	0.0%	0.1%	0.265	0.6%	0.1%	0.0%	0.5%
elmParticle = part->GetDefinition();	0.0%	0.1%	0.619	0.7%	0.2%	0.2%	0.4%
elmKinEnergy = part->GetKineticEnergy();	0.0%	0.0%	0.330	0.1%	0.0%	0.0%	0.0%
elmCrossSection = 0.0;	0.0%	0.0%	0.317	0.1%	0.0%	0.0%	0.1%
G4int i = nDataSetList-1;	0.3%	0.3%	0.583	4.3%	1.7%	1.4%	2.2%
G4int Z = elm->GetZasInt();							
if (elm->GetNaturalAbundanceFlag() &&	0.0%	0.0%	0.676	0.1%	0.0%	0.0%	0.0%
dataSetList[i]->IsElementApplicable(part, Z, mat)) {	0.0%	0.0%	1.021	0.2%	0.3%	0.0%	0.2%
// element wise cross section							
elmCrossSection = dataSetList[i]->GetElementCrossSection(part, Z, mat);	0.0%	0.0%	0.878	0.1%	0.1%	0.0%	0.1%
//G4cout << "Element wise " << elmParticle->GetParticleName()							
// << " xsec(barn)= " << elmCrossSection/barn							
// << " E(MeV)= " << elmKinEnergy/MeV							
// << " Z= " << Z << " AbundFlag= " << elm->GetNaturalAbandancesFlag()							
// <<G4endl;							
} else {							
// isotope wise cross section							
size_t nIso = elm->GetNumberOfIsotopes();							
// user-defined isotope abundances							
const G4double* abundVector = elm->GetRelativeAbundanceVector();							
for (size_t j=0; j<nIso; ++j) {	0.1%	0.2%	0.447	3.0%	0.6%	0.3%	1.1%
if(abundVector[j] > 0.0) {	0.4%	0.5%	0.609	8.1%	2.1%	0.9%	4.5%
const G4Isotope* iso = elm->GetIsotope(j);							
elmCrossSection += abundVector[j]*	0.5%	0.5%	0.706	8.7%	1.8%	1.0%	5.6%
GetIsoCrossSection(part, Z, iso->GetN(), iso, elm, mat, i);	0.9%	1.2%	0.570	15.0%	4.6%	4.0%	9.2%
//G4cout << "Isotope wise " << elmParticle->GetParticleName()							
// << " xsec(barn)= " << elmCrossSection/barn							
// << " E(MeV)= " << elmKinEnergy/MeV							
// << " Z= " << Z << " A= " << iso->GetN() << " j= " << j << G4endl;							
}							
}							
}							
//G4cout << " E(MeV)= " << elmKinEnergy/MeV							
// << "xsec(barn)= " << elmCrossSection/barn <<G4endl;							
return elmCrossSection;	0.0%	0.0%	0.725	0.3%	0.1%	0.1%	0.4%
}	0.2%	0.2%	0.514	3.0%	0.7%	0.3%	1.8%

Bad Speculation in G4PhysicsVector::Interpolation()

186	inline G4double G4PhysicsVector::Interpolation(const std::size_t idx,							
187	const G4double e) const							
188	{							
189	// perform the interpolation							
190	const G4double x1 = binVector[idx];	0.0%	31,487,000,000	0.363	0.7%	0.0%	0.0%	0.3%
191	const G4double dl = binVector[idx + 1] - x1;	0.1%	95,749,000,000	0.531	2.0%	0.3%	0.4%	1.6%
192	// note: all corner cases of the previous methods are covered and eventually							
193	// gives b=0/1 that results in y=y0\y_{N-1} if e<=x[0]/e>=x[N-1] or							
194	// y=y_i/y_{i+1} if e<x[i]/e>=x[i+1] due to small numerical errors							
195	const G4double b = std::max(0., std::min(1., (e - x1) / dl));	0.2%	223,652,000,000	0.754	4.4%	0.7%	3.3%	5.6%
196	G4double res;							
197	if(useSpline) // spline interpolation	0.1%	99,268,000,000	0.697	1.9%	0.4%	0.8%	2.6%
198	{							
199	const G4double os = 0.166666666667; // 1./6.							
200	const G4double a = 1.0 - b;	0.0%	6,624,000,000	0.573	0.2%	0.0%	0.0%	0.2%
201	const G4double c0 = (a * a * a - a) * secDerivative[idx];	0.0%	5,658,000,000	0.760	0.1%	0.0%	0.0%	0.2%
202	const G4double c1 = (b * b * b - b) * secDerivative[idx + 1];	0.9%	511,313,000,000	1.312	11.5%	5.2%	8.6%	30.6%
203	res =							
204	a * dataVector[idx] + b * dataVector[idx + 1] + (c0 + c1) * dl * dl * os;	0.3%	79,189,000,000	2.420	1.6%	1.5%	3.8%	9.1%
205	}							
206	else // linear interpolation							
207	{							
208	const G4double y1 = dataVector[idx];							
209	const G4double y2 = dataVector[idx + 1];							
210	res = y1 + b * (y2 - y1);	0.0%	2,967,000,000	2.822	0.1%	0.0%	0.0%	0.5%
211	}							
212	return res;	0.0%	3,634,000,000	6.095	0.2%	0.0%	0.5%	1.1%
213	}							

Problem is likely due to **`if(useSpline) { … }`**. Recommendation: make useSpline a template parameter.

Bad Speculation in G4PolyhedraSide::Inside() (1)

Source	Clockticks	Instructions Retired	CPI Rate	Locators			
				Retiring	Front-End Bound	Bad Speculation	Back-End Bound
`// Inside`							
`//`							
`EInside G4PolyhedraSide::Inside( const G4ThreeVector& p,`							
`G4double tolerance,`							
`G4double* bestDistance )`							
`{`	0.2%	0.2%	0.703	11.3%	2.5%	7.5%	8.8%
`  //`							
`  // Which phi segment is closest to this point?`							
`  //`							
`  G4int iPhi = ClosestPhiSegment( GetPhi(p) );`	0.0%	0.0%	2.354	1.6%	0.1%	0.7%	1.5%
`  G4double norm;`							
`  //`							
`  // Get distance to this segment`							
`  //`							
`  *bestDistance = DistanceToOneSide( p, vecs[iPhi], &norm );`	0.1%	0.2%	0.449	14.3%	0.4%	0.0%	7.5%
`  //`							
`  // Use distance along normal to decide return value`							
`  //`							
`  if ( (std::fabs(norm) < tolerance) && (*bestDistance < 2.0*tolerance) )`	0.1%	0.0%	1.111	2.7%	5.1%	4.1%	2.0%
`    return kSurface;`	0.0%	0.0%	1.565	1.5%	0.0%	1.6%	0.9%
`  else if (norm < 0)`	0.0%	0.0%	1.102	0.4%	0.6%	1.2%	0.3%
`    return kInside;`							
`  else`							
`    return kOutside;`	0.1%	0.1%	1.076	4.9%	0.0%	4.4%	2.3%
`}`	0.1%	0.1%	1.395	4.3%	0.0%	8.9%	2.2%

Maybe it would be better to rework (avoid branching) or reorder the conditions from most to least probable.

Bad Speculation in G4PolyhedraSide::Inside() (2)

Source	Clockticks	Instructions Retired	CPI Rate	Locators			
				Retiring	Front-End Bound	Bad Speculation	Back-End Bound
//							
G4bool G4PolyPhiFace::InsideEdges(G4double r, G4double z,							
G4double* bestDist2,							
G4PolyPhiFaceVertex** base3Dnorm,							
G4ThreeVector** head3Dnorm)							
{	0.0%	0.1%	0.395	1.2%	0.2%	0.1%	0.4%
G4double bestDistance2 = kInfinity;	0.0%	0.0%	0.000	0.0%	0.0%	0.0%	0.0%
G4bool answer = false;	0.0%	0.0%	0.000	0.0%	0.0%	0.0%	0.0%
G4PolyPhiFaceEdge* edge = edges;	0.0%	0.0%	0.397	0.4%	0.0%	0.0%	0.4%
do // Loop checking, 13.08.2015, G.Cosmo							
{							
G4PolyPhiFaceVertex* testMe;							
//							
// Get distance perpendicular to the edge							
//							
G4double dr = (r-edge->v0->r), dz = (z-edge->v0->z);	0.4%	0.3%	1.125	5.0%	0.1%	1.9%	13.3%
G4double distOut = dr*edge->tz - dz*edge->tr;	0.5%	0.3%	1.063	6.6%	0.4%	2.2%	17.3%
G4double distance2 = distOut*distOut;	0.1%	0.1%	0.473	2.4%	0.0%	0.2%	1.6%
if (distance2 > bestDistance2) continue; // No hope!	0.2%	0.3%	0.461	5.4%	0.0%	0.9%	2.1%
//							
// Check to see if normal intersects edge within the edge's boundary							
//							
G4double q = dr*edge->tr + dz*edge->tz;	0.1%	0.1%	0.491	2.5%	1.9%	0.4%	0.9%

Indirections are expensive. Recommendation: Reconsider the data structures storing properties to avoid pointers.

Indirection in G4SteppingManager::DefinePhysicalStepLength()

Source	Clockticks	Instructions Retired	CPI Rate	Locators			
				Retiring	Front-End Bound	Bad Speculation	Back-End Bound
//							
void G4SteppingManager::DefinePhysicalStepLength()							
//							
{	0.0%	0.1%	0.561	2.2%	3.0%	0.8%	1.0%
// ReSet the counter etc.							
//							
PhysicalStep = DBL_MAX; // Initialize by a huge number	0.0%	0.0%	0.333	0.1%	0.0%	0.0%	0.0%
physIntLength = DBL_MAX; // Initialize by a huge number	0.0%	0.0%	0.424	0.3%	0.0%	0.5%	0.1%
#ifdef G4VERBOSE							
if(verboseLevel>0) fVerbose->DPSLStarted();							
#endif							
// GPIL for PostStep							
//							
fPostStepDoItProcTriggered = MAXofPostStepLoops;	0.0%	0.0%	0.769	0.0%	0.0%	0.1%	0.0%
for(std::size_t np=0; np<MAXofPostStepLoops; ++np)	0.1%	0.0%	0.760	1.8%	0.0%	1.5%	1.0%
{							
fCurrentProcess = (*fPostStepGetPhysIntVector)(np);	0.2%	0.2%	0.581	6.8%	0.0%	3.6%	1.9%
if (fCurrentProcess == nullptr)	0.0%	0.1%	0.601	1.4%	0.0%	0.9%	1.3%
{							
(*fSelectedPostStepDoItVector)[np] = InActivated;							
continue;							
} // NULL means the process is inactivated by a user on fly							
physIntLength = fCurrentProcess->PostStepGPIL(*fTrack, fPreviousStepSize, &fCondition);	0.0%	0.0%	0.789	0.3%	0.0%	0.1%	0.3%

Indirection cannot be removed, but if tracks could be processed in bulk, this line would need to be run less frequently.

Geant4 10.7 in CMSSW

Top functions by self time

Total %	Self	Symbol name
13.41	477.13	G4PolyhedraSide::Intersect(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&, bool, double, double&, double&, CLHEP::Hep3Vector&, bool&)
4.89	173.91	__ieee754_atan2_avx
4.25	151.28	G4Navigator::LocateGlobalPointAndSetup(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const*, bool, bool)
3.02	107.31	G4VEmProcess::PostStepGetPhysicalInteractionLength(G4Track const&, double, G4ForceCondition*)
2.24	79.67	G4PolyhedraSide::Distance(CLHEP::Hep3Vector const&, bool)
2.08	73.95	G4PolyhedraSide::DistanceAway(CLHEP::Hep3Vector const&, G4PolyhedraSide::sG4PolyhedraSideVec const&, double*)
1.73	61.51	G4PolyhedraSide::DistanceToOneSide(CLHEP::Hep3Vector const&, G4PolyhedraSide::sG4PolyhedraSideVec const&, double*)
1.71	60.77	G4SteppingManager::DefinePhysicalStepLength()
1.43	51.04	__tls_get_addr
1.40	49.70	G4DormandPrince745::Stepper(double const*, double const*, double, double*, double*)
1.37	48.83	G4UniversalFluctuation::SampleFluctuations(G4MaterialCutsCouple const*, G4DynamicParticle const*, double, double, double)
1.36	48.45	G4Mag_UsualEqRhs::EvaluateRhsGivenB(double const*, double const*, double*) const
1.32	47.08	G4PolyhedraSide::PhiSegment(double)
1.30	46.39	G4UrbanMscModel::SampleCosineTheta(double, double)
1.20	42.66	G4TouchableHistory::GetVolume(int) const
1.18	41.92	G4Navigator::ComputeStep(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&, double, double&)
1.14	40.41	G4NavigationLevel::operator=(G4NavigationLevel const&)
1.08	38.31	__sin_avx
0.90	32.06	G4PolyhedraSide::GetPhi(CLHEP::Hep3Vector const&)
0.86	30.48	G4CrossSectionDataStore::GetCrossSection(G4DynamicParticle const*, G4Element const*, G4Material const*)
0.85	30.21	G4SteppingManager::Stepping()
0.80	28.35	G4Transportation::PostStepDoIt(G4Track const&, G4Step const&)
0.76	27.15	G4Transportation::AlongStepGetPhysicalInteractionLength(G4Track const&, double, double, double&, G4GPILSelection*)

Full report available at https://dpiparo.web.cern.ch/dpiparo/cgi-bin/igprof-navigator/sim_phase2_g4_107.pp/self

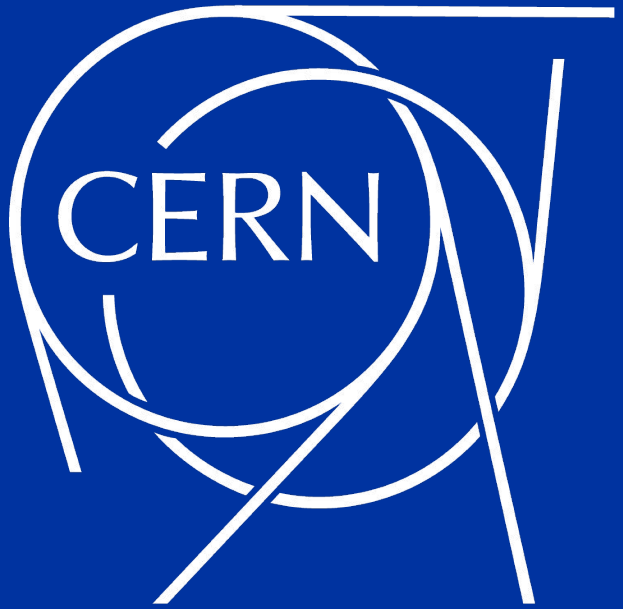
Stepping Overview

Rank	% total	Counts		Paths		Symbol name
		to / from this	Total	Including child / parent	Total	
	93.18	3,314.81	3,466.01	1	1	G4TrackingManager::ProcessOneTrack(G4Track*)
[22]	**93.18**	**30.21**	**3,284.60**	**1**	**1**	**G4SteppingManager::Stepping()**
	53.33	1,897.10	1,897.10	1	1	G4SteppingManager::DefinePhysicalStepLength()
	23.62	840.44	840.44	1	1	G4SteppingManager::InvokePostStepDoItProcs()
	9.96	354.31	354.31	1	1	G4SteppingManager::InvokeAlongStepDoItProcs()
	2.02	71.89	71.89	1	1	SteppingAction::UserSteppingAction(G4Step const*)
	1.59	56.44	56.44	1	1	CaloSD::ProcessHits(G4Step*, G4TouchableHistory*)
	0.53	18.69	42.66	1	15	G4TouchableHistory::GetVolume(int) const
	0.36	12.78	12.78	1	1	G4Region::GetRegionalSteppingAction() const
	0.29	10.15	10.15	1	1	G4SteppingManager::InvokeAtRestDoItProcs()
	0.21	7.47	9.48	1	4	G4TouchableHistory::~G4TouchableHistory()
	0.17	5.92	7.25	1	2	G4StepPoint::operator=(G4StepPoint const&)
	0.15	5.44	5.44	1	1	TkAccumulatingSensitiveDetector::ProcessHits(G4Step*, G4TouchableHistory*)
	0.08	2.96	2.96	1	1	TimingSD::ProcessHits(G4Step*, G4TouchableHistory*)
	0.03	0.96	3.24	1	3	aCountedObjectAllocator()
	0.00	0.06	0.06	1	1	CaloTrkProcessing::ProcessHits(G4Step*, G4TouchableHistory*)
	0.00	0.01	0.01	1	1	MuonSensitiveDetector::ProcessHits(G4Step*, G4TouchableHistory*)

Geometry and Transportation Dominate

Rank	% total	Counts: to / from this	Counts: Total	Paths: Including child / parent	Paths: Total	Symbol name
	38.69	1,376.20	1,897.10	1	1	G4SteppingManager::DefinePhysicalStepLength()
[24]	**38.69**	**27.15**	**1,349.06**	**1**	**1**	**G4Transportation::AlongStepGetPhysicalInteractionLength(G4Track const&, double, double, double&, G4GPILSelection*)**
	22.48	799.57	1,021.33	1	4	G4Navigator::ComputeStep(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const&, double, double&)
	14.61	519.72	519.72	1	1	G4PropagatorInField::ComputeStep(G4FieldTrack&, double, double&, G4VPhysicalVolume*, bool)
	0.45	15.88	92.52	1	4	G4Navigator::ComputeSafety(CLHEP::Hep3Vector const&, double, bool)
	0.12	4.17	4.17	1	1	G4PropagatorInField::FindAndSetFieldManager(G4VPhysicalVolume*)
	0.08	2.92	2.92	1	1	CMSFieldManager::ConfigureForTrack(G4Track const*)
	0.06	2.27	2.27	1	1	G4RKIntegrationDriver<G4MagIntegratorStepper>::GetEquationOfMotion()
	0.05	1.62	1.62	1	1	G4Mag_UsualEqRhs::SetChargeMomentumMass(G4ChargeState, double, double)
	0.04	1.45	42.66	1	15	G4TouchableHistory::GetVolume(int) const
	0.04	1.30	1.30	1	1	G4FieldTrack::G4FieldTrack(CLHEP::Hep3Vector const&, double, CLHEP::Hep3Vector const&, double, double, double, CLHE
	0.00	0.09	0.09	1	1	CMSFieldManager::setDefaultChordFinder()
	0.00	0.04	23.22	1	474	_init
	0.00	0.03	0.03	1	1	CMSFieldManager::setChordFinderForVacuum()
	0.00	0.01	0.01	1	1	CMSFieldManager::setChordFinderForTracker()

It seems that CMS might benefit from a thorough look into optimizations directly in the GDML description of the detector. **Need to understand role of G4PolyhedraSide, and optimize it carefully in Geant4 and/or VecGeom.**


CERN