Vectorization of CMSSW offline software

Dr Patrick Gartung¹ on behalf of the CMS Collaboration

¹ Fermi National Accelerator Laboratory



FERMILAB-POSTER-23-029-CMS-CSAID

Profiling with Intel's OneAPI Toolset

CMSSW uses Intel's Thread Building Blocks (TBB) library. Intel Vtune is used to profile the CMSSW application as it is the only profiler that can stitch together TBB thread stack frames. Intel Advisor is used to identify scalar loops with floating point operations that can potentially be replaced by SIMD instructions – auto-vectorization. Intel's legacy compiler ICC and small vector math library (SVML) can produce more vectorization with the SSE3 instruction set but are not validated for use in CMSSW.

Example of changes to vectorize

An example of changes made to auto-vectorize a loop with GCC and the SSE3 instruction set

RecoLocalTracker/SiPixelRecHits: changes to vectorize loops in VVIObjF::VVIObjF constructor #392	218	dit <> Code 👻
⊱ Merged cmsbuild merged 6 commits into cms-sw:master from gartung:gartung-vviobjf-vectorize 🖓 on Aug 31, 2022		
Q Conversation 41 - Commits 6 F. Checks 0 E Files changed 1		+121 -36
Changes from all commits - File filter - Conversations - Jump to - 😥 -	0 / 1 files viewed	eview changes 👻

jle View Help				4	/storage/local/datal/gruscratch/gartung/CHSSW_11_2_X_2020-07-06-11	100/TimeHemory/offload-reco - Intel Advisor Beta	
🕼 🖄 🖄 🎲 🐻 🕨 Start Su	urvay Analysis 🗸 🔗 📽 🙆			File View Help			
				🕼 🖄 🗃 🕼 🖪 🕨 Start Su	urvey Analysis 🗢 😽 📴 🕐		
Welcome e000 🗶				Welcome e000 X			•
vectorization Inreading	Elapsed time: 567.658 🔆 🕜 Vectorized 🕐 Not Vectorized 🖑 FLTER: (3.99%) IbBRecov Summary 🗞 Survey & Roofine 🎈 Refinement Reports	rtexPrimaryVertexProducer so All Sources 👻	INTEL ADVISOR BET	vectorization inreading	Elapsed time: 567.688 🤾 🕐 Vectorized 💿 Net Vectorized 💿 PLTER: [3.98%] bbRecoVertesPrimaryVertesProduce	er.so 👻 All Sources 👻	INTEL ADVISOR BETA
OFF Batch mode	Vectorization Advisor Vectorization Advisor is a vectorization analysis toolset that lets you identify loops that will benefit m	nost from vector parallelism, discover performance issues preventing from effective vectorization and character	rize your memory vs. vectorization bottlenecks with Advisor Roofline	1. Survey Target	Threading Advisor Threading Advisor Advisor Threading Advisor is a threading design and prototyping tool that lets you analyze, design, tune, and check threading design or	ntions without disruption your normal development	
🕨 Collect 🖿 📴	model automation.			1.1 Find Trip Counts and FLOP	r nameng ranaan ia a anaming aaagir ana proorphing taorana raa yaa amaysa, aaagir, tara, ana anaan anaming aaagir a	prono monour ana apong your norma ao receptions.	
With Callstacks				G Collect 🕅 🖿 🗔	O Drogram matrice		
1. Survey Target	Program metrics Elapsed Time 567.68s Vector Instruction Set SSE3. SSE2. SSE	Number of CPU Threads 7		Trip Counts FLOP	 ⊘ Program metrics Elapsed Time 567.68s Vector Instruction Set \$ 55E3, SSE2, SSE 	Number of CPU Threads 7	
	© Performance characteristics				Performance characteristics		
Mark Loops for Deeper Analysis Select checkboxes in the Survey & Roofline tab to mark loops for other Advisor analyses. There are no marked loops	Metrics Total CPU time	Total 1235.025	•	 Annotate Sources Add Intel Advisor annotations to <u>identify</u> possible parallel tasks and their enclosing parallel sites. Steps to annotate 			
1.1 Find Trip Counts and FLOP	Time in 15 vectorized loops Time in scalar code	▶ 8.115 0.7% 1226.91s 99.3%					
G Collect 🖣 🖿 🕞	Time in scalar code	1226.915		3. Check Suitability	⊘ Per program recommendations		
✓ Trip Counts				🔹 Collect 🕅 🖿 💽	Higher instruction set architecture (ISA) available Consider recompiling your application using a higher ISA. <u>Show more</u>		
FLOP				4. Check Dependencies	conservation grant approximation and a regime to a <u>servation and a</u>		
0 Analyze all loops		Consider rebuilding your application with Intel Compiler 16.0 and higher		💱 Collect 🖿 📘			
2.1 Check Memory Access Patterns					⊙ Top time-consuming loops⑦		
🖡 Collect 🖿 🗽					Consider adding parallel site and task annotations around these time-consuming loops found during Survey analysis. Loop	Self Time ①	Total Time®
🛆 No loops selected	⊘ Per program recommendations				O loop in DAClusterizerInZT_vect:update at DAClusterizerInZT_vect.cc:458	0.210s	21.650s
2.2 Check Dependencies	Higher instruction set architecture (ISA) available				O loop in DAClusterizerinZT_vect::thermalize at DAClusterizerinZT_vect.cc:652	<0.001s	15.950s
🕄 Collect 🖿 🔝	Consider recompiling your application using a higher ISA. Show more				O loop in <u>DAClusterizerInZT_vect:vertices</u> at <u>DAClusterizerInZT_vect.cc:1135</u> O loop in <u>DAClusterizerInZT_vect:update</u> at <u>DAClusterizerInZT_vect.cc:117</u>	<0.001s <0.001s	14.540s 12.860s
🛆 No loops selected					O loop in <u>DAClusterizerinz1_vect: update</u> at <u>DAClusterizerinz1_vect.cc:117</u> O loop in <u>DAClusterizerinZT_vect::vertices at DAClusterizerinZT_vect.cc:1253</u>	<0.001s	12.860s 10.500s
	⊙ Top time-consuming loops⑦						
	Loop	Self Time ①	Total Time [®]		⊙ Suitability and Dependencies analysis data⑦		
	O loop in operator() at DAClusterizerInZT_vect.cc:421	4.010s	4.010s				
	O loop in DAClusterizerInZT_vect::purge at DAClusterizerInZT_vect.cc:793	1.230s	1.230s		A No data available. Collect Suitability or Dependency to see the results.		
	O loop in <u>DAClusterizerInZ_vect::purge</u> at <u>DAClusterizerInZ_vect.cc:686</u>	1.110s	1.110s 1.060s				
	O loop in <u>operator()</u> at <u>DAClusterizerInZT_vect.cc:388</u> O loop in <u>local_exp_list</u> at <u>DAClusterizerInZT_vect.cc:109</u>	1.060s 0.930s	1.060s 4.640s		⊗ Recommendations ⑦		
		0,000	4.0403		Vuse the Intel short vector math library for vector intrinsics loop in DAClusterizerInZT_vect.:fill at DAClusterizerInZT_vect.cr	<u>cc.216</u>	
	⊗ Refinement analysis data ⑦				© Collection details		
	A No data available. Collect MAP or Dependency to see the results.				Collection details		
G Re-finalize Survey	⊗ Recommendations [®]			G Re-finalize Survey	⊘ Platform information		

come e000 X													
Elapsed time: 567.68s 😽 🙆 Vectorized 🗿 Not Vectorized 🧭	FILTE	R: [3.98%] libRecoVe	ertex Primary V	ertexProducer.s	a 👻 All Sources	▼ Loops ▼ All Threads	*					Custo	omize View
Summary 🖏 Survey & Roofline 🛄 Refinement Reports													INTEL ADVISOR
Some target modules do not contain debug information Suggestion: enable debug information for relevant modules.	on												
+ Function Call Sites and Loops		Performance Issues	CPU Time Total Time	Self Time 🗸	Туре	Why No Vectorization?	Vectorized	Loops Gain VL (Ve	Instruction Set Analysis	Data Types	Discussion Location		
区例 [loop in DAClusterizerInZ_vect::purge at DAClusterizerInZ_vec		1990119	1.110s1	1.110s		Compiler lacks sufficie	vector 0	Sain VL (Ve	Iraits	Float64		erinZ vect.cc:686	
Scalar loop	Sc	alar loop											
[Icop in operator() at DAClusterizerInZT_vect.cc:388] Vectorized SSE; SSE2 loop processes Float32; Float64		staniand CCE, CCE3	1.060s I		Vectorized (Body)	- (-)	SSE2	2; 4		Float32;	DAClusterize	rInZT_vect.cc:388	
C [loop in local exp list at DAClusterizerInZT_vect.cc:109]		ctorized SSE; SSE2		0.930s[2(5)			Divisions; Shifts; Shuffles; T	Float32:	DAClusterize	rinZT_vect.cc:109	
Scalar loop with instructions that use SSE; SSE2 regi		alar loop with ins							bitisions, sincs, shames, 1	1100036, 111	Difeitasterize	THE COLOR	
C [loop in DAClusterizerInZT_vect::purge at DAClusterizerInZT_vect.c			0.760s (Vectorized (Body)		SSE2	2; 4		Float32;	DAClusterize	rInZT_vect.cc:778	
Vectorized SSE; SSE2 loop processes Float32; Float64	Ve	ctorized SSE; SSE2				e(s)							
CO[loop in operator() at DAClusterizerInZT_vect.cc:402] Vectorized SSE2 loop processes Float64 data type(s)	Ve	ctorized SSE2 loop	0.700s (Vectorized (Body)		SSE2	2		Float64	DAClusterize	rinZT_vect.cc:402	
C [loop in local_exp_list at DAClusterizerInZ_vect.cc:101]	_	@ 1 Data type conv		0.650s (Divisions; Shifts; Shuffles; T	Float32;	DAClusterize	rinZ_vect.cc:101	
Scalar loop with instructions that use SSE; SSE2 regi		alar loop with ins											
CO [loop in DAClusterizerInZT_vect::split at DAClusterizerInZT_vect.cc:		@ 2 Unoptimized fl		0.600s (Divisions; Square Roots; Ty .	. Float32;	DAClusterize	rInZT_vect.cc:955	
Scalar loop with instructions that use SSE2 registers	150	alar loop with ins	0.440sf			1 Compiler lacks sufficien	CCE2	2:4		Float32:	DAClusterize	rinZ_vect.cc:341	
So [loop in operator() at DAClusterizerInZ_vect.cc:341]			0.430s (Vectorized (Body)	a compiler lacks sufficient.	SSE2	2; 4		Float32;		rinZ_vect.cc:341	
[loop in operator() at DAClusterizerInZ_vect.cc:341]			0.010s(0.010s (Scalar	Compiler lacks sufficient				Float64	DAClusterize	rinZ_vect.cc:341	
urce Top Down Code Analytics Assembly Recommendations	6 Why	No Vectorization?											
: DAClusterizerinZ vect.cc:686 DAClusterizerinZ vect::purge	,	no reconzation											
e					Source						Total Time %	Loop/Function Time	% Traits
<pre>> ppsux_euclietx1 = y.pk_pcitk1 / ty.pk_pcitk1 + inocons ></pre>	1271												
for (unsigned int k = 0; k < nv; k++) { for (unsigned int i = 0; i < nt; ++i) {													
const auto track z = tks.z ptr[i];													
<pre>const auto botrack_dz2 = -beta * tks.dz2_ptr[i];</pre>													
const auto mult_resz = track_z - y.z_ptr[k]; parg_cache[i] = botrack_dz2 * (mult_resz * mult_res	/												

	r uuse reger toou mousuurgenoo	aces an only provertee's term as that it more a successful to the				6	le View Help		oca1/data1/gpuschatch/gantung/Unso		and and				
	View Help														
	🖌 🖄 📴 🎲 💼 🕨 Start Survey Analysis 🗢 🤹 🎏 🕐							0							
	econe econ X					. V	Nelcome e000 %								
						- [👩 Elapsed time: 567.68s 🧩 🧿 Vectorized 📋 Not Vectorized 🦉 F	FILTER: [3.98%] libRecoVertexPrimaryVe	rtexProducer.so 👻 All Sources	▼ Loops ▼ All Threads	*				Customize View
		All Sources V Loops All Threads V			Customize View	<u> </u>	🕎 Summary 🤣 Survey & Roofline 📑 Refinement Reports								INTEL ADVI
					INTEL ADVE	ETI	🗧 🥂 Some target modules do not contain debug information	n							< 1 of 3 🕨
					4 1 of 3 🖡	,									
							7 T = Function Call Sites and Loops	Performance CPU Time	Type	Why No Vectorization?		-	Advanced	Decation	
	Function Call Sites and Loops Performance Issues Total Time Self Time	pe Why No Vectorization? Vector		Advanced	20 Location	Ð	9	local time	Self Time 🗸		Vector Gain VL	. (Ve Traits	Data Types		c:686
			Contraction of the contraction o		DAClusterizerInZ_vect.cc:686		2		1.110s(Scalar	Compiler lacks sumcle			Float64	DACIUSTERIZERINZ_Vect.c	C:606
											SSE2 2; 4	4	Float32;	DAClusterizerInZT_vect.cc	::388
			2; 4	Float32;	DAClusterizerInZT_vect.cc:388					e(s)		Divisions Chifte Chufflers 7	fleet22	D4 Churchesine de 77 une et en	-100
			Divisions: Shifts: Shuffles	s: T Float32:	DAClusterizerInZT_vect.cc:109							Divisions; Shifts; Shuffles; 1	I Float32;	DACIUSterizerInZT_vect.cc	::109
											SSE2 2; 4	4	Float32;	DAClusterizerInZT_vect.cc	
			2: 4	Float32;	DAClusterizerInZT_vect.cc:778										
				Electric 4	Pt Charles for 27, us to as 102						SSE2 2		Float64	DAClusterizerInZT_vect.cc	::402
			2	Float64	DACIUSTERIZERINZ I_Vect. cc:402							Divisions: Shifts: Shuffles: 1	Eloat32	DAClusterizarIn7 vect cc1	101
		calar	Divisions; Shifts; Shuffles	s; T Float32;	DAClusterizerInZ_vect.cc:101							Divisions, sinita, sindines, i		Dictaster terma_vecccc.	101
								2 Unoptimized fl 1.830s	0.600s (Scalar			Divisions; Square Roots; Ty	Float32;	DAClusterizerInZT_vect.cc	:955
			Divisions; Square Roots;	; Ty Float32;	DAClusterizerInZT_vect.cc:955										
			2:4	Elect23	DAChusterizerle7 west se 241										
											5552 2,1	-			
	Ioop in operator() at DAClusterizerinZ_vect.cc:341] 0.010s [0.010s [zalar Compiler lacks sufficient		Float64	DAClusterizerInZ_vect.cc:341										
Building and and all dependences Souther the set of the set							Source Top Down Code Analytics Assembly @Recommendations	Why No Vectorization?							
N South South North Nor	urce Top Down Code Analytics Assembly 💡 Recommendations 🖬 Why No Vectorization?														
	:: DAClusterizerInZ_vect.cc:686 DAClusterizerInZ_vect::purge								Assem	bly				Total Time %	Self Time %
0 relinged (11 + 0) + x + x + 0) { fr (e ppmus_cuchcts; = y.ps_pcrts; / ty.ps_pcrts; / inocursc;;	Source			Total Time % Loop/Function Time % Tra										
0 0)													0.010s (0.010s (
0 (c) (c)	F for (unsigned int $k = 0$: $k < nv$: $k++$) (0.270s (0.270s (
 															0.020s (
															0.250s (
<pre>prcode(1) = bit read_cd2 2 tout[_read_cd2 2 tout[_</pre>							0x2eelc 687 mulsd %xmm3, %xmm0								0.050s (
1 coll english(parg_cabe, peix_cabe, nt); 0 coll english(parg_cabe, peix_cabe, peix															0.010s (
[028206] [0028206] <td< td=""><td>)</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>0.010s (0.190s (</td></td<>)														0.010s (0.190s (
in the find one = 0 in the find one	local_exp_list(parg_cache, peik_cache, nt);														0.050s (
double supp = 0; double supp = 0; const i = 0; i < mt; * *) {	int nUnique = 0:														0.020s (
double max = pmax_check[i] pmax_check[i] + pixerse_rsums[i] 1110.000ms 0x2ee37 666 pix 0x2ee00 <8lock1> Const auto p = y, ph_ctr(k] * pixerse_rsums[i]: 330.020ms 0.999ms 0.999ms 0.02ee37 666 pix 0x2ee00 <8lock1> const auto p = y, ph_ctr(k] * pixerse_rsums[i]: 330.020ms 0.999ms 0.999ms 0.02ee37 666 pix 0x2ee00 <8lock1> isumpix = p; nutrique += ((p = uniquetrikeight_* pmax) & (tks, pi_ptr(i] > 0)) 7 1 : 0; 0.999ms 549.992ms	double sump = 0;													0.220s	0.220s
I for (unique 3) if (int) i=0 i = (it) +1 i +1 i I i 0.000ms (unique 7) 0ACUSTERIZED VECTION OF															
<pre>const auto p = y.pk_ptr[k] * peix_cache[i] * pinverse_zsums[i]: sump *= p: nUnique += ((p > uniquetrkweight_ * pmax) 6 (tks.pi_ptr[i] > 0)) ? 1 : 0; y = y.pk_ptr[k] * pinverse_zsums[i]:</pre>	<pre>@[loop in DAClusterizerInZ_vect::purge at DAClusterizerInZ_vect.cc:686]</pre>				220.002ms 1110.000ms										
<pre>sump += p:</pre>															
<pre>nUnique += ((p > uniquetrkweight_ * pmax) & (tks.pi_ptr(i] > 0)) ? 1 : 0; } if ((nUnique < 2) & (tks.pi_ptr(i] > 0)) ? 1 : 0; sumpin = sump; k = k; } ;</pre>															
<pre>if ((nUnique < 2) 66 (sump < sumpmin)) { sumpsin = sump; k0 = k; } }</pre>	<pre>nUnique += ((p > uniquetrkweight_ * pmax) & (tks.pi_ptr[i] > 0)) ? 1 : 0;</pre>														
3 sumpsin = sump; 4 k0 = k; 5 }															
33 sumpmin = sump: 34 k0 = k; 35 } 36 }	12 if ((nUnique < 2) && (sump < sumpmin)) {														
15) 16)	3 sumpmin = sump;														
	7												Selected ([otal Time]: 0s	

<pre>const float xp[9] = {9.29, 2.47, .89, .36, .15, .07, .03, .02, 0.0}; const float xq[7] = {.012, .03, .08, .26, .87, 3.83, 11.0}; float h_[7]; float q, u, h4, h5, h6, q2, d, ll, ul, rv; int lp, lq, k, l, n; // Make sure that the inputs are reasonable } q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c3[n]; c4[0] = 0.f; float s[n]; s1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c3[n]; c3[0] = 0.f; float c3[n]; c4[0] = 0.f; float x[n]; x1[0] = 0.f; float x[n]; x1[0] = 0.f; float x[1]; x1[0] = 0.f; float x[1]; x1[0] = 0.f; float x[1]; x1[0] = 0.f; float x1[n]; x1[0] = 0.f; float x1[n</pre>	
<pre>const float xq[7] = {.012, .03, .08, .26, .87, 3.83, 11.0}; float h_[7]; float q, u, h4, h5, h6, q2, d, ll, ul, rv; int lp, lq, k, l, n; // Make sure that the inputs are reasonable } q = -1.; q2 = 2.; float x[n]; x1[0] = 0.f; float c1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float x[n]; x1[0] = 0.f; float c4[n]; c4[0] = 0.f; float x[n]; x1[0] = 0.f; float x</pre>	
<pre>float h_[7]; float q, u, h4, h5, h6, q2, d, l1, u1, rv; int lp, lq, k, l, n; // Make sure that the inputs are reasonable g = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float s[n]; s[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x1[k] = mega_ * k; x1[k] = n6 * x[k]; } for (k = 1; k < n; ++k) { (k = 1; k < n; +</pre>	
<pre>int lp, lq, k, l, n; // Make sure that the inputs are reasonable } q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float c4[n]; c4[0] = 0.f; float c1[n]; c4[0] = 0.f; float c1[n]; x1[0] = 0.f; float x1[n]; x1[0] = 0.f; for (k = 1; k < n; ++k) {</pre>	
<pre>// Make sure that the inputs are reasonable } q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float c1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = mega_* k; x1[k] = h6 *x[k]; } for (k = 1; k < n; ++k) { </pre>	
<pre>} q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float c4[n]; c4[0] = 0.f; float c1[n]; s[0] = 0.f; float c1[n]; c[0] = 0.f; float x1[n]; x1[0] = 0.f; float x1[n]; x1[0] = 0.f; float x2[n]; x1[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_* k; x1[k] = h6 * x[k]; } </pre>	
<pre>q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c1[n]; c1[0] = 0.f; float x11[n]; x12[0] = 0.f; float x12[n]; x12[0] = 0.f; float x2[n]; x12[0] = 0.f; for (k = 1; k < n; ++k) {</pre>	
<pre>q = -1.; q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c1[n]; c1[0] = 0.f; float x11[n]; x12[0] = 0.f; float x12[n]; x12[0] = 0.f; float x2[n]; x12[0] = 0.f; for (k = 1; k < n; ++k) {</pre>	
<pre>q2 = 2.; float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s10] = 0.f; float x11[n]; x11[0] = 0.f; float x12[n]; x12[0] = 0.f; float x12[n]; x12[0] = 0.f; for (k = 1; k < n; ++k) { x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float x[n]; x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>x[0] = 0.f; float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float s[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float x1[n]; x1[0] = 0.f; float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float s[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float c1[n]; c1[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>cl[0] = 0.f; float c2[n]; c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>c2[0] = 0.f; float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float c3[n]; c3[0] = 0.f; float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float c4[n]; c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>c4[0] = 0.f; float s[n]; s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>s[0] = 0.f; float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float c[n]; c[0] = 0.f; float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>float xf1[n]; xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>xf1[0] = 0.f; float xf2[n]; xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>xf2[0] = 0.f; for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>for (k = 1; k < n; ++k) { x[k] = omega_ * k; x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>x1[k] = h6 * x[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>} for (k = 1; k < n; ++k) {</pre>	
}	
<pre>for (k = 1; k < n; ++k) { c1[k] = vdt::fast_logf(x[k]) - c1[k];</pre>	
}	
<pre>for (k = 1; k < n; ++k) { vdt::fast_sincosf(x1[k], c3[k], c4[k]);</pre>	
<pre>xf1[k] = kappa * (beta2 * c1[k] - c4[k]) - x[k] * c2[k];</pre>	
<pre>xf2[k] = x[k] * c1[k] + kappa * (c3[k] + beta2 * c2[k]) + t0_ * x[k]; }</pre>	
for $(k = 1; k < n; ++k)$ {	
<pre>vdt::fast_sincosf(xf2[k], s[k], c[k]); }</pre>	
<pre>float d1[n]; t1[0] = 0 fr</pre>	
d1[0] = 0.f; if (mode_ == 0) {	
<pre>for (k = 1; k < n; ++k) { d1[k] = q * d * omega_ * vdt::fast_expf(xf1[k]);</pre>	
q = -q;	
<pre>} for (k = 1; k < n; ++k) {</pre>	
l = n - k;	
a_[l - 1] = d1[k] * c[k]; b_[l - 1] = -d1[k] * s[k];	
}	
<pre>} else { for (k = 1; k < n; ++k) {</pre>	
d1[k] = q * d * vdt::fast_expf(xf1[k]) / k;	
q = -q;	
q = -q; }	
<pre>} for (k = 1; k < n; ++k) {</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k;</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; }</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; } for (k = 1; k < n; ++k) {</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; }</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; } for (k = 1; k < n; ++k) { l = n - k; b_[l - 1] = d1[k] * c[k]; }</pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; } for (k = 1; k < n; ++k) { l = n - k; b_[l - 1] = d1[k] * c[k]; } for (k = 1; k < n; ++k) { l = n - k; </pre>	
<pre>} for (k = 1; k < n; ++k) { l = n - k; a_[l - 1] = d1[k] * s[k]; } for (k = 1; k < n; ++k) { l = n - k; b_[l - 1] = d1[k] * c[k]; } for (k = 1; k < n; ++k) {</pre>	
+ + +	<pre>+ l = n - k; + a_[l - 1] = d1[k] * s[k]; + } + for (k = 1; k < n; ++k) { + l = n - k;</pre>

Screen caps of intel Advisor showing Vectorization Advisor pane, Threading Advisor pane, Survey pane with Source view of loop with timing, and Survey pane with Assembly view of loop with timing.

133	}	179	}
134 -	-		
135	} // VVIObjF	180	} // VVIObjF
136		181	
137	//	182	//
	*********		***************************************
	*************		*******

Screencap of partial diff from Github pull request for CMSSW https://github.com/cms-sw/cmssw/pull/39218/files

CMSSW multi microarchitecture releases and performance

For production, CMSSW is optimized for the SSE3 instruction set, the lowest common instruction set. Optimizing for higher instruction sets and vector widths enables more loops to be vectorized potentially increasing performance. The CMSSW build system was enhanced to produce three sets of libraries. These are compiled with the GCC march flags sse3, haswell and skylake-avx512. The library set is chosen automatically unless overridden by an environment variable.

Micro-architecture	GEN-SIM	DIGI-HLT	RECO	Micro-architecture	GEN-SIM	DIGI-HLT	RECO
default	0.079 ev/sec per thread	0.087 ev/sec per thread	0.082 ev/sec per thread	default	0.056 ev/sec per thread	0.053 ev/sec per thread	0.045 ev/sec per thread
haswell	0.090 ev/sec per thread	0.099 ev/sec per thread	0.077 ev/sec per thread	haswell	0.051 ev/sec per thread	0.053 ev/sec per thread	0.053 ev/sec per thread
skylake-avx512	0.080 ev/sec per thread	0.087 ev/sec per thread	0.078 ev/sec per thread	skylake-avx512	0.052 ev/sec per thread	0.052 ev/sec per thread	0.052 ev/sec per thread

Throughput results on a 50% loaded node for a Run 3 workflow for micro-architectures 11th Gen Intel Core i7 i11700 @ 2.50Ghz

Throughput results on a 100% loaded node for a Run 3 workflow for micro-architectures 11th Gen Intel Core i7 i11700 @ 2.50Ghz

The use of higher vector widths can cause the boost clock to be lowered for thermal management¹. For the 50% loaded RECO process, the throughput is lower for higher vector widths. This indicates that the boost clock was used but lowered. For the 100% loaded RECO process, the throughput is higher for higher vector widths. This indicates that the boost clock was not used and the improvement from higher vector widths can be seen. ¹https://cvw.cac.cornell.edu/vector/performance_turbo

This document was prepared by the CMS Collaboration using the resources of the Fermi National Accelerator Laboratory (Fermilab), a U.S. Department of Energy, Office of Science, HEP User Facility. Fermilab is managed by Fermi Research Alliance, LLC (FRA), acting under Contract No. DE-AC02-07CH11359,



