

INTEL® ADVISOR AND ROOFLINE MODEL

Part of Intel[®] Parallel Studio XE

Roadmap Notice: All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Contacts

Advisor Support Mail List vector.advisor@intel.com

Zakhar Matveev <u>zakhar.a.matveev@intel.com</u> Intel Advisor Product Architect Kirill Rogozhin <u>kirill.rogozhin@intel.com</u> Intel Advisor Project Manager Egor Kazachkov <u>egor.kazachkov@intel.com</u> Intel Advisor Senior Developer

Optimization Notice

Copyright © 2015, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others. Intel Confidential



What is Intel® Advisor



Cache Simulator and MAP

Site Location			Strides Distribu	Access Pattern			
Iloop in ComputeTimeStep			80%/0%/2	20%	Mixed strides		
[loop in pricePath_Core at			92% / 0% /	8%	Mixed strides		
<							
Memory Access Patterns Report				Dependencies Report 💡 Recomme			
ID	•	Stride	Туре		Source		
± P53	44	1	Unit strid	le	ch_4_v	253.cpp:76	
⊞ P1	₩		Gather st	Gather stride		253.cpp:76	
± P52	14	1	Unit strid	le	ch_4_v	253.cpp:218	

Python API

import advisor

project = advisor.open_project(sys.argv[1])
data = project.load(advisor.SURVEY)

roofs = data.get_roofs(4, advisor.RoofsStrategy.MULTI_THREAD¹

- for roof in roofs:
 - # memory roofs
 - if 'bandwidth' in roof.name.lower():

bandwidth = roof.bandwidth / math.pow(10, 9) # conve print '{} {:.0f} GB/s'.format(roof.name, bandwidth) # compute roofs

else:

bandwidth = roof.bandwidth / math.pow(10, 9) # conv print '{} {:.0f} GFLOPS'.format(roof.name, bandwidth

Threading prototyping



Optimization Notice



What is Intel[®] Advisor - Collectors



Intel Confidential

Optimization Notice

Collections vs Analysis

Analysis	Collections
Vectorization (basic)	Survey + Trip Counts
Vectorization (advanced)	As above + MAP + Dependencies
Roofline (CARM)	Survey + Trip Counts with FLOP
Roofline (Integrated)	Survey + Trip Counts with FLOP and Cache Simulator
Threading	Survey + Suitability + Dependencies
Custom Analysis (Python API)	Depends
Mix and match as you wish	More data come with a cost





VECTORIZATION

5 Steps to Efficient Vectorization

Intel® Advisor – Vectorization Advisor

1.	1. Compiler diagnostics + Performance Data + SIMD efficiency information					2. Guidance: detect problem and recommend how to fix it							and t			
	ction Call h loopInit a h loopInit a h runCFora h runCRaw	Sites and Loops at LCALSSuite.com at LCALSSuite.com IIILambdaLoops a Loops at runCRat	Self Time 0.016s1 0.016s1 0.072s1 0.578s1	Vectorized Loops Vect Efficiency ▼ G AVX 75% 2. AVX 75% 2. AVX 75% 2. AVX 75% 2. AVX 75% 2.	rsis Data T Float64; Float64; Float64;	All Advisor-detectable issues: C++ I Fortran Recommendation: Add data padding The tip count is not a multiple of vector length. To fix: Do one of the following: SSUE: Ineffective peeled/remail to present							led/remainder			
± ^C (loop in ± ^C (loop in ± ^C (loop in ± ^C (loop in	n runOMPI n runOMPI n runARaw n runAFora	RawLoops\$omp\$ RawLoops\$omp\$ Loops at runARa IllambdaLoops a	p 0.953s) p 1.953s) v 0.734s) e 0.578s	3. Tri	3. Trip Counts + FLOP: understand utilization, parallelism granularity & overheads											
								age	Call 5712	Count 000	Self GFLOPS	Self Al		Add dat		
Garle	4. N	1emor	у Асс		rns A	nalysis	124; 1	; 13;	5.	Loop	o-Carrie	d Dep	bend	lenc	xy Ar	nalysis
් [lo ර [lo ර [lo	op in (op in (ComputeTi pricePath_(imeStep Core at	. 80% / 0% / 92% / 0% /	20%	Mixed strides Mixed strides	m ; ;	Pro ID P1 P2	blems Q Q Q	Type Parallel s Read afte	nges site information er write dependency	Site Name site2 site2	Sources dqtest2.cp dqtest2.cp	op op	Modules dqtest2 dqtest2	State Vot a problem New
Memory Access Patterns Report Dependencies Report P Recomment ID Type Source				nenda	P3 P4 P5 P6	0 0 0	Write afte Write afte Write afte Write afte	er write dependency er write dependency er write dependency er read dependency	site2 site2 site2 site2	dqtest2.cp dqtest2.cp dqtest2.cp dqtest2.cp	99 99 99 99	aqtest2 dqtest2 dqtest2 dqtest2 dqtest2	Re New Re New Re New			
 ₱ P53 ₱ P1 ₱ P52 	₩ ₩	1	Unit strid Gather s Unit strid	de tride de	ch_4_v2 ch_4_v2 ch_4_v2	253.cpp:76 253.cpp:76 253.cpp:218		P7	٩	Write aft	er read dependency	site2	dqtest2.cp	op; idle.h	dqtest2	R New

Optimization Notice



1. Compiler diagnostics + Performance Data + SIMD efficiency information

	C INT	Vectori	zed Loops	Instruction Set Analysis							
Self Time Vectorized Loops Distruction Set Analysis © [loop in looplnit at LCALSSuite.coc 0.0161 AVX 72% 2.99x 4 Divisions; Type C Floe © [loop in looplnit at LCALSSuite.coc 0.0161 AVX 72% 2.99x 4 Divisions; Type C Floe © [loop in looplnit at LCALSSuite.coc 0.0161 AVX 72% 5.99x 4 Divisions; Type C Floe © [loop in runCForallLambdaLoops a 0.672s1 AVX; 70% 5.60x 2; 4; 8 Extracts; FMA; Ty Floe © [loop in runOMPRawLoopsSomp5; 0.953s1 AVX; 70% 5.60x 2; 4; 8 Extracts; FMA; Ty Floe © [loop in runOMPRawLoopsSomp5; 1.953s1 AVX 62% 2.75x 4 Floa Floa E© [loop in runARawLoops at runARaw 0.734s1 AVX2 67% 2.67x 4 Blends; Divisions; Floa E© [loop in runAForallLambdaLoops a 0.578s1 AVX2 67% 2.67x 4 Blends; Divisions; Floa E© [loop in runAForallLambdaLoops a 0.578s1 AVX2	Data T										
🖽 🖱 [loop in loopInit at LCALSSuite.cxx:	0.016s l	AVX	75%	2.99x	4	Divisions; Type C	Float64				
🖽 🖱 [loop in loopInit at LCALSSuite.cox	0.016s l	AVX	75%	2.99x	4	Divisions; Type C	Float64				
🖽 🖱 [loop in runCForallLambdaLoops a	0.672s1	AVX;	70%	5.60x	2; 4; 8	Extracts; FMA; Ty	Float64				
🖽 🖱 [loop in runCRawLoops at runCRav	0.578s1	AVX;	70%	5.60x	2; 4; 8	Extracts; FMA; Ty	Float64.				
🖽 🖱 [loop in runOMPRawLoops\$omp\$p	0.953s I	AVX	69%	2.75x	4	FMA	Float64				
🖽 🖱 [loop in runOMPRawLoops\$omp\$p	1.953s I	AVX	68%	2.74x	4		Float64				
	0.734s I	AVX2	67%	2.67x	4	Blends; Divisions;	Float32;				
🛨 🖱 [loop in runAForallLambdaLoops a	0.578s I	AVX2	67%	2.67x	4	Blends; Divisions;	Float32;				
+ Binary Analysis											

Vector Efficiency: All The Data In One Place

My "performance thermometer"

Evention Coll Sites and Lance	Call Times	Vectori	zed Loops		\gg	Instruction Set Analysis
+ - Function Call Sites and Loops	Self Time	Vect	Efficiency 🔺	Gain	VL (Traits
🗄 🗗 [loop in runCForallLambdaLoops at runCForallLa	0.734s I	AVX;	26%	2.11x	4; 8	Extracts; Inserts; Type Conversions
H Ioop in runCRawLoops at runCRawLoops.cx:704	0.625s1	AVX;	26%	2.11x	4; 8	Extracts; Inserts; Type Conversions
	2.703s 0	AVX2	<mark>_31</mark> %	2.50x	4; 8	FMA; Inserts; Permutes; Unpacks
	2.609s I	AVX2	<mark>_31</mark> %	2.50x	4; 8	FMA; Inserts; Permutes; Unpacks
H Ioop in runOMPRawLoops\$omp\$parallel@135 at	0.453s I	AVX2	45%	1.80x	4	Blends; Divisions; FMA; Masked Stores; Square Roots
🛨 🝊 [loop in runAForallLambdaLoops at runAForallLa	0.234s1	AVX2	45%	1.82x	4	Blends; Divisions; FMA; Masked Stores; Square Roots



- Auto-vectorization: affected <3% of code
 - With moderate speed-ups
- First attempt to simply put #pragma omp simd:
 - Introduced slow-down
- Look at Vector Issues and Traits to find out why
 - All kinds of "memory manipulations"
 - Usually an indication of "bad" access pattern

Survey: Find out if your code is "under vectorized" and why

Optimization Notice



Details for any piece of code

ø	Elapsed time: 1462:35s 👩 Vectorized 👩 Not Vectorized 🖉																	
FILTER	TER: All Modules 👻 All Sources 👻 Loops 👻 All Threads 👻 INTEL ADVISOR 2018																	
🗒 Su	Summary 🇞 Survey & Roofline 🛄 Refinement Reports																	
8	Vectorized Loops									~	Trip Cou	ints 🗵	FLOPS		^			
PF	± -	Function	Call Sites and Loop	Self IIn	ne	¥ Performance iss	ues		Vect	Efficiency 🕶	Gain	VL (Com	Average	Call Count	Self GFLO	PS Self Al	
I.	+ 🖰 I	loop in runC	MPRawLoops\$om	sp 0.953	ls I	💡 1 Ineffective pee	led/remainder lo	oop(s)	. AVX	69%	2.75x	4	3.87x	14; 27; 2	113232000	48.2470	0.12500	
	+ 🖰 I	loop in runC	MPRawLoops\$om	sp 1.953	ls I	💡 1 Ineffective pee	led/remainder lo	oop(s)	. AVX	68%	2.74x	4	<3.08x	27; 2; 2; 2	8176000; 8	15.1001	0.01880	
	±0	[loop in runA	RawLoops at runA	Ra 0.73	4sl	💡 3 Ineffective pe	eled/remainde	r loo	AVX2	67%	2.67x	4	2.66x	6128; 3	2045000;	5.711	0.10232	2
	+ 🖰 (loop in runA	ForallLambdaLoop	at 0.578	3s I	💡 3 Ineffective pee	led/remainder lo	oop(s)	. AVX2	67%	2.67x	4	2.66x	6128; 3	2045000; 2	7.2551	0.10232	
	+ 🖰 I	loop in runC	MPRawLoops\$om	sp 1.578	Bs I	💡 3 Ineffective pee	led/remainder lo	oop(s)	. AVX2	66%	2.62x	4	2.66x	2; 110; 2	113225000;	26.8451	0.12934	
	+ 🖰	loop in runB	RawLoops at runBR	aw 2.437	7s I	💡 3 Ineffective pee	led/remainder lo	oop(s)	. AVX2	64%	2.58x	4	2.57x	6128; 3	1840000; 1	1.5381	0.15299	\sim
	<			> <													:	>
Sour	e	Top Down	Code Analytics	Assembly	💡 R	Recommendations	Why No Vec	torizati	on?									
File: r	unAR		x:150 runARawLoo															
Line						Source						Total	Time	% 1.00	on/Eunction Ti	me %	Traits	^
150	Ξ		for (Index	type i=0) • i	i/len • i++) J						Tota	0.078s	70 200	0°	734c	mana	
150	_ <	0 [loop :	in runARawLoops	at runA	RawL	00ps.cxx:1501						-	0.0703		0.			
		Vecto	orized AVX; AVX	2; FMA 1	oop	processes Float	64; Int32; U	Int32;	; UInte	4 data typ	e(s) ai	n						
		No l	oop transformat	ions app	lied	-												
		(100p :	in runARawLoops	at runA	RawL	oops.cxx:150]												
		Scal	ar remainder lo	op with	inst	ructions that u	se AVX regis	ters										
		No l	oop transformat	ions app	lied													
		() [loop]	in runARavLoops	at runA	RavL	oops.cxx:150]												
		Scal	ar peeled loop	[not exe	cute	d] with instruc	tions that u	se AVI	X regis	ters								
		No 1	oop transformat	ions app	lied	I						_						
151			Real_type	e q_tilde	: ;												Blends	
152																		
153			if (delv	:[i] > 0.	.0) {	[0.047s					
154			q_tile	ie = 0. ;									0.062s					
155			}							Coloritoria / Tot	-1 Time)		0.070-					
										Selected (101	ai Time)		0.0785					
	~																	Contraction 1

Optimization Notice

1. Compiler diagnostics + Performance Data + SIMD efficiency information

🖂 🗔 . Evention: Cell Sites and Leaves	Calif Times	Vectori	zed Loops		Instruction Set Analysis		
Punction Call Sites and Loops	Seir rime	Vect	Efficiency 🕶	Gain	VL (Traits	Data T
王の [loop in loopInit at LCALSSuite.coc		AVX	75%	2.99x	4	Divisions; Type C	Float64;
亜〇 [loop in loopInit at LCALSSuite.cxx		AVX	75%	2.99x	4	Divisions; Type C	Float64;
王の [loop in runCForallLambdaLoops a		AVX;	70%	5.60x	2; 4; 8	Extracts; FMA; Ty	Float64;
亜岱 [loop in runCRawLoops at runCRav		AVX;	70%	5.60x	2; 4; 8	Extracts; FMA; Ty	Float64
亜岱 [loop in runOMPRawLoopsSompSp	0.953s I	AVX	69%	2.75x	4	FMA	Float64
王ण [loop in runOMPRawLoops\$omp\$p	1.953s I	AVX	68%	2.74x	4		Float64;
±の [loop in runARawLoops at runARav	0.734s I	AVX2	67%	2.67x	4	Blends; Divisions;	Float32; F
±の [loop in runAForallLambdaLoops a		AVX2	67%	2.67x	4	Blends; Divisions;	Float32; F

2. Guidance: detect problem and recommend how to fix it

All Advisor-detectable issues: C++ | Fortran

Recommendation: Add data padding

The trip count is not a multiple of vector length. To fix: Do one of the following:

- Increase the size of objects and add iterations so the trip count is a multiple of vector length.
- Increase the size of static and automatic objects, and use a compiler option to add data padding.

Windows* OS	Linux* OS
/Qopt-assume-safe-padding	-gopt-assume-safe-padding



An of some source loop tertations are not executing in the <u>loop body</u>. Improve performance by moving source loop iterations from <u>peeled</u>/remainder loops to the loop body.

Add data padding

Get Specific Advice For Improving Vectorization

Intel® Advisor – Vectorization Advisor



All Advisor-detectable issues: C++ | Fortran



All or some <u>source loop</u> iterations are not executing in the <u>loop body</u>. Improve premainder loops to the loop body.



Add data padding

The trip count is not a multiple of vector length. To fix: Do one of the following:

- · Increase the size of objects and add iterations so the trip count is a multiple of vector length.
- Increase the size of static and automatic objects, and use a compiler option to add data padding.

Advisor shows hints to move iterations to vector body.

by moving source loop iterations from peeled/

Optimization Notice



1. Compiler diagnostics + Performance Data + SIMD efficiency information

	C. 16 T.	Vectori	zed Loops			Instruction Set Analysis		
+ - Function Call Sites and Loops	Self Lime	Vect	Efficiency 🕶	Gain	VL (Traits	Data	
① [loop in loopInit at LCALSSuite.cxc		AVX	75%	2.99x	4	Divisions; Type C	Float	
王の [loop in loopInit at LCALSSuite.cxc		AVX	75%	2.99x	4	Divisions; Type C	Float	
王の [loop in runCForallLambdaLoops a		AVX;	70%	5.60x	2; 4; 8	Extracts; FMA; Ty	Float	
王の [loop in runCRawLoops at runCRav			-	•				
王の [loop in runOMPRawLoops\$omp\$p	0.953s I		ЗТ	rin	Col	unts + Fl	0	
王の [loop in runOMPRawLoops\$omp\$p	1.953s I		5.1	קיי	201			
± ⁽ 」[loop in runARawLoops at runARav	0.734s I			r	bara	allelism	gra	
± ⁽ 」(Ioop in runAForallLambdaLoops a								
					<u>с п</u>	en 11	Tr	
			+ - Fu	nctior	i Call	Sites and Loops	Δ.	

2. Guidance: detect problem and recommend how to fix it

All Advisor-detectable issues: <u>C++</u> | <u>Fortran</u> Recommendation: Add data padding The <u>trip count</u> is not a multiple of <u>vector length</u>. To fix: Do one of the following:

Trip Counts + FLOP: understand utilization, parallelism granularity & overheads

Evention Coll Sites and Lance	Trip Counts	≫	FLOPS	
+ - Function Call Sites and Loops	Average	Call Count	Self GFLOPS	Self Al
☑ ^[] [loop in runOMPRawLoops\$omp	111	5712000	427.516	0.22794
	124; 1; 13;	46816000;	204.298 📖	0.17103

U 1 SUE: Ineffective peeled/remainder op(s) present

or some <u>source loop</u> iterations are not ecuting in the <u>loop body</u>. Improve informance by moving source loop iterations im <u>peeled</u>/<u>remainder</u> loops to the loop idy.

Add data paddin

Critical Data Made Easy Loop Trip Counts Knowing the time spent in a loop is not enough! Trip Counts Function Call Sites and Loops Self Time Type + -Min Average Max Call Count [loop in runOMPRawLoops\$omp\$r] 4.190s 0 Vectorized+Threaded (Body; Peeled; Remainder) 2; 110; 2 112590000 ... 1; 17; 1 3; 111; 3 [] [loop in runOMPRawLoops\$om] 3.768s 0 Remainder+Threaded (OpenMP) 2 1 3 1125900000 0.406s l Iloop in runOMPRawLoops\$om Vectorized (Body)+Threaded (OpenMP) 110 17 111 12320000 [loop in runOMPRawLoops\$om] [] 0.016s1 Peeled+Threaded (OpenMP) 2 1 3 880000 1.1 Find Trip Counts and FLOP Check G Collect Find trip counts for actual trip each part of a loop Trip Counts counts **FLOP**

Optimization Notice

Precise Repeatable FLOP Metrics

Intel[®] Advisor – Vectorization Optimization

- FLOPS by loop and function
- All recent Intel processors

- Instrumentation (count FLOP) plus sampling (time with low overhead)
- Adjusted for masking with AVX-512 processors

E E Function Coll Steeland Loope	Calf Times	Vectori	zed Loops		FLOPS		
+ - Function Call Sites and Loops	Self Time	Vect	Efficiency	Gain	VL (Self GFLOPS	Self Al
🖃 🗗 [loop in runOMPRawLoops\$omp\$p	1.984s I	AVX;	100%	4.30x	4	204.298	0.17103
☑ ⁽⁵] [loop in runOMPRawLoops\$om]	1.469s I	AVX2			4	398.921	0.17574
☑ ⁽⁵] [loop in runOMPRawLoops\$om]	0.078s l	AVX			4	20.633 0	0.06250
☑ [™] [loop in runOMPRawLoops\$om	0.141s I					13.1521	0.06250
☑ [™] [loop in runOMPRawLoops\$om	0.234s l					12.7971	0.14315
☑ [™] [loop in runOMPRawLoops\$om	0.063s l					0.1041	0.06250
🗄 🗗 [loop in runOMPRawLoops\$omp\$p	1.406s I	AVX2	52%	1.05x	2	107.057 🖿	0.22428
🗄 🖱 [loop in runOMPRawLoops\$omp\$p	1.172s I	AVX	81%	3.22x	4	63.354 🛛	0.07500



1. Compiler diagnostics + Performance Data + SIMD efficiency information

	C IN T	Vectorized Loops						
+ - Function Call Sites and Loops	Self Time	Vect	Efficiency 🕶					
⊞ ^[] [loop in loopInit at LCALSSuite.cxx		AVX	75%					
⊞ ⁽⁵] [loop in loopInit at LCALSSuite.cxx:		AVX	75%					
団 [loop in runCForallLambdaLoops a		AVX;	70%					
団 [loop in runCRawLoops at runCRav								
団 [loop in runOMPRawLoopsSompSp	0.953s1		3 T	r				
団 [loop in runOMPRawLoops\$omp\$p	1.953s I		3.1	ł.				
⊞ ⁽⁾ [loop in runARawLoops at runARav	0.734s I							
⊞© [loop in runAForallLambdaLoops a								

2. Guidance: detect problem and recommend how to fix it

All Advisor-detectable issues: <u>C++</u> <u>Fortran</u>
Recommendation: Add data padding
The $\underline{trip\ count}$ is not a multiple of $\underline{vector\ length}.$ To fix: Do one of th following:

3. Trip Counts + FLOP understand utilization, parallelism granularity & overheads

The function Coll Changed Lange	Trip	Counts		FLOPS	
+ - Function Call Sites and Loops	Aver	ige	Call Count	Self GFLOPS	Self Al
☑ ^[] [loop in runOMPRawLoops\$omp	111		5712000	427.516	0.22794
TT/5 (I	424.4	. 12.	46016000	204 200	0.17102
		, 10, m	40010000,	204,290	0.17105

ISSUE: Ineffective peeled/remainde

All or some <u>source loop</u> iterations are not executing in the <u>loop body</u>. Improve performance by moving source loop iterations from <u>peeled/remainder</u> loops to the loop body.

Add data padding

3. Memory Access Patterns Analysis

Site Lo	cation			Strides Distribution 🔺		Access Pattern	
🖱 [la	oop in (ComputeT	imeStep	80% / 0% / 2	20%	Mixed strides	
🖸 [le	oop in p	oricePath_(Core at	92% / 0% /	8%	Mixed strides	
<							
Memo	ory Acc	ess Patterr	ns Report	Dependencies	s Report	@ Recommenda	
ID	•	Stride	Туре		Source	•	
± P53	4-4	1	Unit strid	le	ch_4_v	253.cpp:76	
⊞ P1	₩ł		Gather st	ride	ch_4_v	253.срр:76	
± P52	44	1	Unit strid	le	ch_4_v	253.cpp:218	

Improve Vectorization

Memory Access pattern analysis



Optimization Notice



Advisor Memory Access Pattern (MAP):

know your access pattern

Unit-Stride access

for (i=0; i<N; i++)
A[i] = C[i]*D[i]</pre>

Constant stride access

for (i=0; i<N; i++)
point[i].x = x[i]</pre>

Variable stride access

for (i=0; i<N; i++)
A[B[i]] = C[i]*D[i]</pre>



Optimization Notice

Find vector optimization opportunities Memory Access pattern analysis

Site Location	Strides Distribution 🔺	Access Pattern	Max. Site Footprint				
[loop in ComputeTimeStepKer	80% / 0% / 20 <mark>%</mark>	Mixed strides	2KB				
[loop in pricePath_Core at ch_3.	. 92% / 0% / 8% 📕	Mixed strides	1KB				
Memory Access Patterns Report D	ependencies Report	Recommendation	s				

All Advisor-detectable issues: C++ | Fortran

Recommendation: Refactor code with detected regular stride access patterns

The Memory Access Patterns Report shows the following regular stride access(es):

Variable					
block 0x2e23c404b80 allocated at cache aligned allocator.	cpp:196 Invariant				

See details in the Memory Access Patterns Report Source Details view.

To improve memory access: Refactor your code to alert the compiler to a regular stride access. Sometimes, it might be beneficial to use the ipo/Qipo compiler option to enable interprocedural optimization (IPO) between files.



1. Compiler diagnostics + Performance **Data + SIMD efficiency information**

	C INT:	Vectorized		
+ - Function Call Sites and Loops	Self Time	Vect	Eff	
⊞ ^[] [loop in loopInit at LCALSSuite.cxc		AVX		
⊞ ⁽⁵] [loop in loopInit at LCALSSuite.cxx:		AVX		
団 [loop in runCForallLambdaLoops a		AVX;		
団 [loop in runCRawLoops at runCRav				
団 [loop in runOMPRawLoops\$omp\$p	0.953s I			
団 [loop in runOMPRawLoops\$omp\$p	1.953s I			
⊞ ^{(I} [loop in runARawLoops at runARav	0.734s I			
⊞© [loop in runAForallLambdaLoops a				

				recomment
		Instruction Set Analysis		
	VL (Traits	Data T	
	4	Divisions; Type C	Float64;	Recommendation: Add data padding
	4	Divisions; Type C	Float64;	The trip count is not a multiple of vector length. To fix: Do one of fellowing:
	2; 4; 8	Extracts; FMA; Ty	Float64;	lonowing.
)	Со	unts + Fl	OP	understand utilization,

+ - Function Call Sites and Loops

4. Memory Access Patterns Analysis

Site Location	Strides Distribution \blacktriangle	Access Pattern
O [loop in ComputeTimeStep	80% / 0% / 20%	Mixed strides
🖱 [loop in pricePath_Core at	92% / 0% / 8%	Mixed strides
<		

ID	•	Stride	Туре	Source
⊞ P53		1	Unit stride	ch_4_v253.cpp:76
. ₽52	•	1	Unit stride	ch_4_v253.cpp:218

2. Guidance: detect problem and recommend how to fix it

3. Tri parallelism granularity & overheads

5. Loop-Carried Dependency Analysis

Problems and Messages								
ID	٩	Туре	Site Name	Sources	Modules	State		
P1	0	Parallel site information	site2	dqtest2.cpp	dqtest2	✔ Not a problem		
P2	٥	Read after write dependency	site2	dqtest2.cpp	dqtest2	Rew New		
P3	٨	Read after write dependency	site2	dqtest2.cpp	dqtest2	Rew New		
P4	0	Write after write dependency				R New		
P5	٥	Write after write dependency	site2	dqtest2.cpp	dqtest2	Rew New		
P6	٥	Write after read dependency	site2	dqtest2.cpp	dqtest2	Rew New		
P7	٥	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	Rew New		

ENABLING VECTORIZATION



Vactor locues	Solf Timo-	Total Time	Tuno	Vectorized	Loops		≫
vector issues	Sell Time	iotai rime	туре	Vector ISA	Efficiency	Gain	VL (V
	10.507sI	22.989s1	Scalar				
2 Possible inef	1.762s	3.190s	Vectorized Ver	AVX512	73%	5.84x	8
					4 5 6 /		-

Optimization Notice

Is It Safe to Vectorize?

Loop-carried dependencies analysis verifies correctness

	Summary	🏷 Survey & Roofline	Refinemen	t Repor	ts	
ROOFL	+ - F	unction Call Sites and Lo	ops	۵	Self Time	Why No Vectorization?
Ī	🛛 🖱 [loo	p in maxPriceCore at ch_	3_1_5_kernel_m	\checkmark	34.565s 📖	vector dependence prevents vectorization
	🛛 🕛 [loo	p in maxPriceCore at ch_	3_1_5_kernel_m	\checkmark	26.027s 🔲	vector dependence prevents vectorization
	🛛 🛛 🖸 [loo	p in OptionDecision at ch	_4_v253.cpp:18	\checkmark	0.360s l	vector dependence prevents vectorization
	🛛 🕛 [loo	p in _10 <lambda0> at ch</lambda0>	_3_1_5_kernel_r	\checkmark	0.016s l	vector dependence prevents vectorization



Optimization Notice



Correctness – Is It Safe to Vectorize?

Loop-carried dependencies analysis

🖉 Check for loop-carried dependencies in your application 🖿										
🤗 Summary 🛛 😂 Survey Report 🍅 Refinement Reports 💧 Annotation Report 🛛 🍿 Suitability Report										
Site Name Site Function Site Info Loop-Carried Dependencies						Strides Distribution Access Pattern				
loop_site_6	main	main.cpp:13	RAW:1	A WAR:1 A WAW:	:1 919	6/ 0% / 9% 📕 Mixed stride	5			
Detected										
						elected				
					de	eleclea Poendencie	24			
					de	ependencie	es			
Memory Ac	cess Patterns Repo	t Correctne	ess Report		de	ependencie	S			
Memory Ac Problems an	icess Patterns Repo id Messages	rt Correctne	ess Report	/	de	ependencie	?S			
Memory Ac Problems an ID 🔊	ccess Patterns Repo id Messages Type	rt Correctne	ess Report Site Name	Sources	de	state	2S			
Memory Ac Problems an ID & P1 II	ccess Patterns Repo d Messages Type Parallel site inform	rt Correctne	ess Report Site Name oon site_6	Sources main.cpp	Modules test_1.exe	state Vot a problem	?S			
Memory Ac Problems an ID & P1 II P3 &	cess Patterns Repoi d Messages Type Parallel site inform Read after write d	t Correctne	ess Report Site Name oon site_6 oon_site_6	Sources main.cpp crtexe.c; main.cpp	Modules test_1.exe	State Not a problem Not a problem	25			
Memory Ac Problems an ID & P1 I P3 & P4 &	cess Patterns Repo d Messages Type Parallel site inform Read after write d Write after write d	t Correctne	ess Report Site Name oop_site_6 oop_site_6 oop_site_6	Sources main.cpp crtexe.c; main.cpp crtexe.c; main.cpp	Modules test_1.exe test_1.exe test_1.exe	State New New	25			

Write after read dependency: Code Locations										
ID	Description	Source	Function	Module	State					
EX.	7 Read	🖹 main.cpp:22	main	test_1.exe	Re New					
4	20	k += a[9];								
	21	k *= a[8];								
	22	k -= a[7];								
	23	k += a[6];								
	24	k *= a[5];								
ΞX	8 Read	🖹 main.cpp:23	main		B. M					
1	21	k *= a[8];		C	 Business states in a state state 					
	22	k -= a[7];		Source	e lines with Read and					
	23	k += a[6];								
				Write a						

Received recommendations to force vectorization of a loop:

- Mark-up loop and check for REAL dependencies 1.
- Explore dependencies with code snippets 2.

In this example 3 dependencies were detected:

- RAW Read After Write
- WAR Write After Read
- WAW Write After Write

This is NOT a good candidate to force vectorization!

Optimization Notice



Data Dependencies – Tough Problem

Is it safe to force the compiler to vectorize?

Elapsed time: 3.64s 🙆 Vectorized 🗿 Not Vector	zed 🖉 MKL FILTER: All Modules	▼ All Sources ▼								
🗒 Summary 🗞 Survey & Roofline 📲 Refinement Reports										
Site Location 💌	Loop-Carried Dependencies	Recommendations								
[loop in maxPriceCore at ch_3_1_5_kernel_max.cpp:103]	RAW:1									
[loop in maxPriceCore at ch_3_1_5_kernel_max.cpp:103]	No dependencies found									
[loop in dS0MaxKernel at ch_5_2.cpp:708]	🛆 Potential WAR:1 🛆 Potential WAW:1									
[loop in dS0MaxKernel at ch_4.h:74]	No dependencies found									
[loop in dS0MaxKernel at ch_4.h:66]	No dependencies found									
[loop in _10 <lambda0> at ch_3_1_5_kernel_max.cpp:243]</lambda0>	No dependencies found	1 Assumed dependency present								
<										
Memory Access Patterns Report Dependencies Report 💡 Recommendations										

All Advisor-detectable issues: C++ | Fortran

Recommendation: Enable vectorization

The Dependencies analysis shows there is no real dependency in the loop for the given workload. Tell the compiler it is safe to vectorize using the restrict keyword or a <u>directive</u>:

Directive	Outcome				
#pragma simd or #pragma omp simd	Ignores all dependencies in the loop				
#pragma ivdep	Ignores only vector dependencies (which is safest)				

Optimization Notice

ROOFLINE

Questions to answer with Roofline: for your loops / functions

Am I doing well? How far am I from the peak? (do I utilize hardware well or not?) Where is the final bottleneck?

(where will be my limit after all optimizations?) Long-term ROI, optimization strategy





Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others. Intel Confidential



Automated Roofline Chart Generation in Advisor - CARM



Summarized memory-compute efficiency picture for the application

Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others. Intel Confidential



Roofline picture

Chart configuration

Roof configuration



Switch to grid represenation

Optimization Notice



Integrated Roofline. What is my current limit?

Performance is limited by minimum of intercepts (L2, LLC, DRAM, CPU)

In this case: by DRAM

Integrated Roofline for one kernel with traffic for all memory levels

Peak Flop/s



Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others. Intel Confidential

NEW*: Selecting Integer, Float or Mixed operations



Optimization Notice

Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others. Intel Confidential

Integer Operations in Survey Grid and Loop Analytics

Use settings button on the "Compute Performance" column

	🖹 Summary 🤌 Survey & Roofline 📲 Refiner	nent Re	eports					//			
II.			👝 🛓 💡 Performance	Self Time	Total Time	Tupo	Compute Performance			🔺 🔍 🔺	
н			Issues			rype	Self GINTO	Self GINTOPS Self GINTOP		Show Floating-Point Operations	
н	Ioop in EXACT_RHS at exact_rhs.		🛾 🔋 3 Unoptimize	. 0.000s	0.000s1	Inside vectorized	103.707 🗆	0.002	 Show I 	nteger Operations	
II.	Icop in Z_SOLVE at z_solve.f:332			0.040s I	3.699s 🛛	Scalar	14.382 🛛	0.575	Show N	/lixed Compute Operation	ons
II.	Solution Soluti Solution Solution Solution Solution Solution Solution S			0.050s I	3.369s 🛙	Scalar	11.499 0	0.575	0.050s	3.3695	
II.	Icop in Y_SOLVE at y_solve.f:326			0.070s l	3.690s 🛙	Scalar	8.2210	0.575	0.070s	3.690s	
II.	☑ [loop in COMPUTE_RHS at rhs.f:273]			0.010s I	0.821s)	Scalar	2.7971	0.028	0.010s	0.821s	
II.	☑ ⁽⁵] [loop in X_SOLVE at x_solve.f:384]			0.250s I	0.250s I	Scalar	1.1691	0.292	0.250s	0.250s	
II.	☑ ⁽⁵] [loop in Z_SOLVE at z_solve.f:396]			0.280s I	0.280s I	Scalar	1.0441	0.292	0.280s	0.280s	
II.	☑ ^[] [loop in COMPUTE_RHS at rhs.f:134]			0.090s I	0.090s I	Vectorized (Body)	1.0001	0.090	0.090s	0.090s	
J.	A										►
	Source Ton Down Code Analytics Assemt	dv 😡	Recommendations	🖬 Why No Vi	ectorization?]					
в	Source Top Down Code Analytics Assemb	", -	Recommendations	 Any NO 0. 	Seconzación:						
	Loop in EXACT_RHS at exact_rhs.f.139					Average Trip Co	nts: [®] 64	Use type expanded	selecto d Loop	r in 🤤	
	Inside vectorized AVX; AVX2 Instruction Se	f Tota < 0. f Self	.001s :time			Statistics for INT(And Data Transfe FLC	DP DP	Analytics	Section	Self Total	
L						owroa [®] INT	'OP	Per 100p	7 00000- 00	4 40000- 07	
L	Static Instruction Mix Static	immaŋ	6				01	0.00172	7.00000e-09	4.48000e-07	
L	Dynamic Instruction Mill	c Sumn	nary			GINTOPS INT	+FLOAT	03.70670	0.00042	0.02698	
Memory 41% (4182272, 17)						INTAI	~	0.03977	1.61667e-07	0.00001	
L	Compute 1/% (1/)	Mask Utilization All Operations									
▶ Mixed 15% (1476096, 6)						L1 Gb		0.04330	1.76000e-07	0.00001	
	Uner 21% (2100170, 11)					L1 Gb/s [®] 2607.48278					
	CPU Total Time					Elapsed Time			0.00002s		
	0,0,		- r m-								

Optimization Notice

Use the rest of the Advisor

Optimization Notice

A few words about callstacks

Available in current product version, does not work with hierarchical roofline yet

Optimization Notice

In a few words

Using Intel[®] Advisor, you can

- Collect the data for the Hierarchical and Integrated Roofline
- Analyze the roofline picture
- Focus on data you are interested in
- Compare roofline for different runs
- Share roofline results
- and more

PYTHON API

Python API

- Small library and Python module to access all Advisor-collected data. Actively used internally at Intel.
- About 200 metrics for each loop/function + about 500 instruction mix metrics.
- Supports:
 - Python 2.7
 - Linux: external and internal Python
 - Windows: internal Python only (advixe-python)
- See examples at <advisor_install_dir>/pythonapi/examples

Optimization Notice

Intel Confidential

36

Python API - example

```
import advisor
project = advisor.open_project(sys.argv[1])
data = project.load(advisor.SURVEY)
roofs = data.get_roofs(4, advisor.RoofsStrategy.MULTI_THREAD)
for roof in roofs:
   # memory roofs
   if 'bandwidth' in roof.name.lower():
        bandwidth = roof.bandwidth / math.pow(10, 9) # converting to GByte/s
        print '{} {:.0f} GB/s'.format(roof.name, bandwidth)
   # compute roofs
   else:
        bandwidth = roof.bandwidth / math.pow(10, 9) # converting to GFLOPS
        print '{} {:.0f} GFLOPS'.format(roof.name, bandwidth)
```


Read more

Intro on [integer] roofline usage:

https://software.intel.com/en-us/articles/a-brief-overview-of-integer-roofline-modeling-in-intel-advisor

Short intro on Python API usage:

https://software.seek.intel.com/LP=18275

Optimization Notice Copyright © 2017, Intel Corporation. All rights reserved. *Other names and brands may be claimed as the property of others.

Intel Confidential

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2016, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804