

# VECTORIZE OR DIE Tutorial

### Legal Notice

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

[BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Inside, Core Inside, i960, Intel, the Intel logo, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, the Intel Inside logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel Sponsors of Tomorrow., the Intel Sponsors of Tomorrow. logo, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, InTru, the InTru logo, InTru soundmark, Itanium, Itanium Inside, MCS, MMX, Pentium, Pentium Inside, skoool, the skoool logo, Sound Mark, The Journey Inside, vPro Inside, VTune, Xeon, and Xeon Inside] are trademarks of Intel Corporation in the U.S. and other countries.

\*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Bluetooth is a trademark owned by its proprietor and used by Intel Corporation under license.

Intel Corporation uses the Palm OS® Ready mark under license from Palm, Inc.

Copyright © 2015, Intel Corporation. All rights reserved.

# INTRODUCTION

The scope and challenge of Vectorization

## Challenge

The process of modernizing todays software is a huge problem but also a great opportunity for Intel. Vectorization in particular is a daunting challenge for customers. In order for customers to have any chance of fully utilizing today's latest hardware they need to thread and vectorize their code. But not all threading or vectorization designs are worthwhile. How do you choose which designs to implement without disrupting ongoing development?

## Performance is a Proven Game Changer

It is driving disruptive change in multiple industries



### Protecting buildings from extreme events

Sophisticated mechanics simulations are performed to identify innovative ways to protect infrastructure from extreme events, such as natural disasters.



### Solving Austin, Texas's traffic problem

Running advanced traffic simulations to improve the models used to plan infrastructure and traffic control changes



### New possible treatments for Parkinson's

Extensive calculations performed at supercomputer helped researchers to learn more about the protein structure's evolution

Click on a picture for details



### The "Free Lunch" is over, really Processor clock rate growth halted around 2005



all the potential performance



# Moore's Law Is Going Strong

### Hardware performance continues to grow exponentially



"We think we can continue Moore's Law for at least another 10 years."

Intel Senior Fellow Mark Bohr, 2015



## **Changing Hardware Impacts Software**

More cores  $\rightarrow$  More Threads  $\rightarrow$  Wider vectors

									inside XEON PHI
	Intel <sup>®</sup> Xeon <sup>®</sup> processor 64-bit	Intel <sup>®</sup> Xeon <sup>®</sup> processor 5100 series	Intel <sup>®</sup> Xeon <sup>®</sup> processor 5500 series	Intel <sup>®</sup> Xeon <sup>®</sup> processor 5600 series	Intel <sup>®</sup> Xeon <sup>®</sup> processor code-named Sandy Bridge EP	Intel <sup>®</sup> Xeon <sup>®</sup> processor code-named Ivy Bridge EP	Intel <sup>®</sup> Xeon <sup>®</sup> processor code-named Haswell EP	Intel® Xeon Phi™ coprocessor Knights Corner	Intel <sup>®</sup> Xeon Phi™ processor & coprocessor Knights Landing <sup>1</sup>
Core(s)	1	2	4	6	8	12	18	61	60+
Threads	2	2	8	12	16	24	36	244	
SIMD Width	128	128	128	128	256	256	256	512	

\*Product specification for launched and shipped products available on ark.intel.com. 1. Not launched or in planning.

### High performance software must be both:

- Parallel (multi-thread, multi-process)
- Vectorized



# **INTEL® ADVISOR XE**

**VECTORIZATION OPTIMIZATION AND THREAD PROTOTYPING** 



### **Auto-Vectorization**

SIMD – Single Instruction Multiple Data

- Scalar mode
  - one instruction produces one result

### SIMD processing

- with SSE or AVX instructions
- one instruction can produce multiple results

for (i=0;i<=MAX;i++)
c[i]=a[i]+b[i];</pre>





## Vector Instructions are Dramatically Faster

Multiple arithmetic operations with a single instruction



• These instructions are also referred to as Single Instruction Multiple Data (SIMD instructions)



### Intel<sup>®</sup> Advanced Vector Extensions (Intel<sup>®</sup> AVX)





### Don't use a single Vector lane!

Un-vectorized and un-threaded software will under perform





### Permission to Design for All Lanes

Threading and Vectorization needed to fully utilize modern hardware





### **Untapped Potential Can Be Huge!**

Threaded + Vectorized can be much faster than either one alone



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. For more information go to <a href="http://www.intel.com/performance">http://www.intel.com/performance</a> <u>Configurations for</u> <u>Binomial Options SP</u> at the end of this presentation



# Data-Driven Threading Design

### Intel<sup>®</sup> Advisor XE – Thread Prototyping

Have you:

- Tried threading an app, but seen little performance benefit?
- Hit a "scalability barrier"? Performance gains level off as you add cores?
- Delayed a release that adds threading because of synchronization errors?

Breakthrough for threading design:

- Quickly prototype multiple options
- Project scaling on larger systems
- Find synchronization errors before implementing threading
- Separate design and implementation -Design without disrupting development



#### Add Parallelism with Less Effort, Less Risk and More Impact



# Data Driven Vectorization Design

Intel<sup>®</sup> Advisor XE – Vectorization Advisor

### Have you:

- Recompiled with AVX2, but seen little benefit?
- Wondered where to start adding vectorization?
- Recoded intrinsics for each new architecture?
- Struggled with cryptic compiler vectorization messages?

### Breakthrough for vectorization design

- What vectorization will pay off the most?
- What is blocking vectorization and why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?

More Performance Fewer Machine Dependencies

🖉 Threading and Vectorization Survey 🖬 🛛 🛛 🛛 Intel Advisor XE 2016											
🌳 Summary 🛭 😂 Survey Re	eport  🦞 S	uitability Rep	ort 🍅 Co	rrectnes	s Report 🍃	Memory Acc	ess P	atterns			
Filter by Loop Type Vectori	zed Not V	/ectorized	Fil	lter by S	ource (All	·	Filt	ter by Module [Al]	~ <b>«</b>		
Function Call Sites and	Calf Times	Takal Times	Memory	Compi	ler Vectoriz	ation	«	Marked and Lange	$\gg$		
Loops	Self Time	iotai fime	analysis	Loop Туре		▲ Gain Estima	ate				
🛛 🔽 [loop at mmult_se	10.040s	10.040s		Vecto	rized	2.19727		SSE2			
▷ [loop at mmult_serial.cp	0.000s	10.100s		Scalar				SSE2			
▷ [loop at mmult_serial.cp	0.000s	10.100s		Scalar							
▷ [loop inlibc_start_mai	0.000s	10.100s	Scalar								
<u></u>									>		
Top Down Source Ass											
Function Call Sites and	Total Time		Hot		Vector	Location			<b>«</b>		
Loops	%	Total Time	Self Time	Loops	Loops	Source Loc	Mod	iule			
⊽Total	100.0%	10.100s	0s								
▼libc_start_main											
⊽ of [loop in _libc_st	100.0%	10.100s	0s				libc-	2.12.so			
マmain	100.0%	10.100s	0s			mmult_seri	1_m	imult_serial			
⊽ of [loop at mm	100.0%	10.100s	0s	5		mmult_seri	1_m	mult_serial			
⊽o [loop at m…	100.0%	10.100s	0s	6	SSE2	mmult_seri	1_m	imult_serial			
⊽multiply_d	100.0%	10.100s	0.0600s			mmult_seri	1_m	imult_serial			
▷ _ [loop	99.4%	10.040s	10.0400s	6	SSE2	mmult_seri	1_m	mult_serial			

## The Right Data At Your Fingertips

### Get all the data you need for high impact vectorization





### **4** Steps to Efficient Vectorization

### Intel<sup>®</sup> Advisor XE – Vectorization Advisor

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

E-mation Coll Stream diamage	Self	Total		•	Compiler Vectorizat	ion
Punction Call Sites and Euopsik	Time	Time		¥	Loop Туре	Why No Vectorization?
⊞ [loop in runCForallLambdaLoops]	0.094:	0.094s			Scalar	vector dependence prevents vector.
⊞ [loop in runCForallLambdaLoops]	0.140:	3.744s			Scalar	inner loop was already vectorized
■ V [loop in std::_Complex_base <double.struct_c_double_complex>::i</double.struct_c_double_complex>	0.031	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 Peeled loop; loop stmts were reordered	data ty	npe(s)	havi	ng Di	visions; Square	Roots operations
Elloop in std::basic_string <char,struct <char="" std::char_traits="">,class std::allo</char,struct>	0.000	544.0			Scalar	nonstandard loop is not a vectoriza.
[loop in std::basic_string <char, <char="" std::char_traits="" struct="">, class std::allo</char,>	0.000:	544.0			Scalar	nonstandard loop is not a vectoriza.
₪ [loop in std::num_put <char,class st<="" std::ostreambuf_iterator<char,struct="" td=""><td>0.000:</td><td>0.234s</td><td></td><td></td><td>Scalar</td><td>nonstandard loop is not a vectoriza.</td></char,class>	0.000:	0.234s			Scalar	nonstandard loop is not a vectoriza.

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

#### 2 Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at Vector Essentials. Utilizing Full Vectors...

Recommendation: Align memory access Projected maximum performance gain: High Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

#### float "array; array = (float \*)\_mm\_malloc(ARRAY\_SIZE\*sizeof(float), 32);

- // Somewhere else \_\_assume\_aligned(array, 32);
  // Use array in loop

#### **3. Loop-Carried Dependency Analysis**

#### 4. Memory Access Patterns Analysis

							Site Name	e Site Function	Site Info	Loop-Carried Dependencies	s Strides Distribution	Acce	ss Pattern
Prol		and Messages					loop_site_	203 runCRawLoop	runCRawLoops.coc1063	RAW:1	No information ava	ilable No in	formation availabl
ID		R	Cite Manue	Courses	Madulaa	Chata	loop_site_	139 runCRawLoop	s runCRawLoops.coc622	No information available	39% / 36% / 25	Mixe	d strides
U		туре	Site Name	Sources	Modules	State	loop_site_	160 runCRawLoop	s runCRawLoops.coc925	No information available	100%/0%/0%	All u	nit strides
P1	0	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem	Memory	Access Patterns	Correctness Report				
P2	0	Read after write dependency	site2	dqtest2.cpp	dqtest2	Rew New	ID 🤻	🕏 Stride 🕶		Туре	Source	Modules	Alignment
P3	0	Read after write dependency	site2	dqtest2.cpp	dqtest2	Rew New	■P22 [	0; 0; 1		Unit stride	runCRawLoops.cxc637	Icals.exe	
P4	0	Write after write dependency		dqtest2.cpp	dqtest2	R New	636		p[ip][0] += y[12	+32];			
P5	0	Write after write dependency	site2	dqtest2.cpp	dqtest2	Rew New	637 638		p[ip][1] += z[j2 i2 += e[i2+32];	+32];			
P6	0	Write after read dependency	site2	datest2 cpp	datest2	New .	639		j2 += f[j2+32];	1			
		tinte attrice a dependency	U.C.	adcoot	aquoun		⊞P23	0;0		Unit stride	runCRawLoops.cxc638	Icals.exe	
P7	. 😲	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	New New	🗆 P30	-1575; -63; -26;	-25; -1; 0; 1; 25; 26; 63; 2164	801 Variable stride	runCRawLoops.cxc628	Icals.exe	
							626		11 4= 64-1;				
							627		j1 4= 64-1;				
							1020		p(1p)(2) += b()1	1(++1)			

80

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

Function Coll State and Leaves	Self	Total		0	Compiler Vectorizat	tion
Punction Call sites and Ecops	Time	Time	٠	¥	Loop Туре	Why No Vectorization?
⊞ [loop in runCForallLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vector
⊞ [loop in runCForallLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
■ V [loop in std::_Complex_base <double.struct_c_double_complex>::i</double.struct_c_double_complex>	0.031s	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 Peeled loop; loop stmts were reordered	data ty	npe(s)	havi	ng Di	visions; Square	Roots operations
⊞ [loop in std::basic_string <char,struct <char="" std::char_traits="">,class std::allo</char,struct>	0.000s	544.0			Scalar	nonstandard loop is not a vectoriza
Iloop in std::basic_string <char, <char="" std::char_traits="" struct="">, class std::allo</char,>	0.000s	544.0			Scalar	nonstandard loop is not a vectoriza
∃ [loop in std::num_put <char,class st<="" std::ostreambuf_iterator<char,struct="" td=""><td>0.000s</td><td>0.234s</td><td></td><td></td><td>Scalar</td><td>nonstandard loop is not a vectoriza</td></char,class>	0.000s	0.234s			Scalar	nonstandard loop is not a vectoriza

### Efficiently Vectorize your code Intel Advisor XE – Vectorization Advisor

Summary	📕 Where should I	Where should I add vectorization and/or threading parallelism? 🗖										KE 20	16
Etapsed time: 54.44;       Vectorized       Not Vectorized       FILTER:       All Modules       All Sources       Vectorized Loops       Vectorized Loops         Function Call Sites and Loops       Image: Control of Contof Control of Control of Control of Contol of	🤗 Summary 🦉 Survey	Report 🍅 Refi	nement Reports 💧 Annota	tion Report	🖞 Suitabilit	y Report							
Function Call Sites and Loops $\bigvee$ Vector issues       Self Time $\bigvee$ Total Time $\bigcap_{Counts}$ $\bigcup_{Cop}$ Type $\bigcup_{Why No Vectorization?$ $\bigvee$ Vectorized Loops $\bigvee$ Vector. Efficiency       Vector. I         10 [loop at idualgo.hd/47d0 in stdt:r       0.17051       0.17051       0.17051       Counts       Collapse       Vectorization? $\bigvee$ Vector       Fficiency       Vector         10 [loop at idualgo.hd/47d0 in stdt:r       0.17051       0.17051       12/4       Collapse       Collapse       AVX       400%         10 [loop at loopsticpp:2449 in s       0.17051       0.17051       0.20261       4       4       Image: Collapse       AVX       400%       4         10 [loop at loopsticpp:2499 in s       0.02051       0.02051       12       Vectorization possible but       4         10 [loop at loopsticpp:2390 in s2/2       11 High vector register       0.15051       12       Expand       Expand       AVX       400%       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8       8	Elapsed time: 54.44s V	ectorized Not	: Vectorized 🖉 👘 FILTE	R: All Modul	es 🗸	All Source	es 🗸 🗸						Q,
rank_column       etcom       bit initial point       count       coupy type       within the count and the count initial point       vector:       vector: <td>Function Call Sites and Loo</td> <td></td> <td>@ Vester laures</td> <td>Salf Times</td> <td>Tatal Times</td> <td>Trip 🗵</td> <td>l con Turco</td> <td>When No Vestavization?</td> <td>Vectori</td> <td>zed Loop</td> <td>ps</td> <td></td> <td>^</td>	Function Call Sites and Loo		@ Vester laures	Salf Times	Tatal Times	Trip 🗵	l con Turco	When No Vestavization?	Vectori	zed Loop	ps		^
a:○ [loop at slight, add Add in stdirt]	Function Call Sites and E00	ibs <b>G</b>	A ASCIDLIZZOS	Seir Time ♥	Total Time	Counts	Loop Type	why NO vectorization:	Vecto	Efficie	ncy	Vector	L
Image: Second	i> 🖱 [loop at stl_algo.h:474	0 in std::tr		0.170s I	0.170s I		Scalar	non-vectorizable loop in	i				
10       0.150:1       0.150:1       12       Vectorized (Body)       AVX       4         10       0.020:1       0.020:1       0.020:1       4       Remainder       4         10       0.000:1       0.020:1       0.020:1       4       Remainder       4         10       0.000:1       0.170:1       0.170:1       0.170:1       0.020:1       0.20:1       4         10       0.000:1       0.160:1       0.160:1       0.160:1       12       Expand       Expand       AVX       4         10       0.000:1       0.150:1       0.150:1       125:4       Expand       Expand       AVX       98%       8         10       0.000:1       0.150:1       125:4       Expand       Expand       AVX       98%       8         10       0.000:1       0.150:1       125:4       Expand       Expand       AVX       40%       4         10       0.000:1       0.150:1       0.150:1       12:5       Expand       Expand       AVX       4       4         10       0.000:1       0.150:1       0.150:1       12:5       Expand       AVX       4       4       4         10       Expand	🗆 🖳 [loop at loopstl.cpp:24	449 in s234_]		0.170s I	0.170s I	12; 4	<u>Collapse</u>	<u>Collapse</u>	AVX	~100	6	4	
10       0.020s1       0.020s1       4       Remainder       4         10       0.020s1       0.020s1       0.020s1       500       Scalar       Vectorization possible but       4         10       0.020s1       0.020s1       0.020s1       0.020s1       0.020s1       Vectorization possible but       4         10       0.020s1       0.020s1       0.020s1       0.020s1       0.020s1       Vectorization possible but       4         10       0.020s1       0.020s1       0.020s1       0.020s1       0.020s1       Vectorization possible but       4         10       0.020s1       0.020s1       0.020s1       0.020s1       0.020s1       Vectorization possible but       4         10       0.020s1       0.020s1       0.020s1       12       Expand       Expand       AVX       S00%       8         10       0.010s1       0.050s1       0.010s1       1050s1       12       Expand       Expand       AVX       100%       4         10       0.010s1       0.050s1       0.010s1       0.050s1       4       Scalar       Vector dependence preve       Vector dependence preve       Vector idependence preve       Vector idependence preve       Imagendenc	🗈 🛄 [loop at loopstl.cpp	0:2449 in s 🗌		0.150s l	0.150s l	12	Vectorized (Body)		AVX			4	
i> 0 [loop at loopstl.cpp:3509 in s2.7]       0.170s1       0.170s1       0.170s1       0.010s1       Expand       Expand       AVX       ≤00         i0 [loop at loopstl.cpp:3509 in s2.79]       Q 2 [ineffective peeled/rem]       0.150s1       0.150s1       125;4       Expand       Expand       AVX       ≤00       0.00%       4         i0 [loop at loopstl.cpp:3509 in s2.79]       Q 2 [ineffective peeled/rem]       0.150s1       0.150s1       125;4       Expand       Expand       AVX       ≤00       6         i0 [loop at loopstl.cpp:3509 in s2.79]       Q 1 Assumed dependency       0.150s1       0.150s1       12       Expand       Expand       AVX       ≤00%       4         i0 [loop at sti_numeric.h:247 in std       Q 1 Assumed dependency       0.150s1       0.150s1       49       Scalar       Vector dependence preve       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V       V	i> 🝊 [loop at loopstl.cpp	0:2449 in s 🗌		0.020s I	0.020s I	4	Remainder						
Iloop at loopstLcpp:3509 in s2 I High vector register 0.160s1 0.160s1 12 Expand AVX 60% 8   Iloop at loopstLcpp:3509 in s2 I High vector register 0.150s1 0.150s1 125;4 Expand Expand AVX 60% 8   Iloop at loopstLcpp:3209 in s2 I Loop at stal_numeric.h:247 in std I Loop at s	i> 🖱 [loop at loopstl.cpp:79	900 in vas_] 🗌		0.170s l	0.170s l	500	Scalar	vectorization possible bu	t			4	
Image: Source interprises in s279       Image: Source interprises in s273       Image: Source interprises in s273       Image: Source interprises interp	🗄 😉 [loop at loopstl.cpp: 3]	509 in s2	I High vector register	0.160s	0.160s	12	Expand	Expand	AVX	~6 <mark>9%</mark>		8	
Image: Source in the set of the set	🗄 🛄 [loop at loopstl.cpp:38	391 in s279_]		0.150s l	0.150s l	125; 4	Expand	Expand	AVX	~9 <mark>6%</mark>		8	
<pre>i&gt; © [loop at stl_numeric.h:247 in std] ♀ 1 Assumed dependency 0.150:1 49 Scalar ♀ vector dependence preve</pre>	🗄 🛄 [loop at loopstl.cpp:62	249 in s414_]		0.150s l	0.150s l	12	Expand	Expand	AVX	~100	6	4	
Yop Down       Source       Loop Assembly       Assistance       Recommendations       Compiler Diagnostic Details         File:       Source       Total Time       %       Loop Time       %         3504       forttime_(stl);       Source       Total Time       %       Loop Time       %         3505       i_l = *ntimes;	i> 🖱 [loop at stl_numeric.h:	:247 in std 🗌		0.150s I	0.150s l	49	Scalar	vector dependence preve					
Top Down       Source       Loop Assembly       Assistance       Recommendations       Compiler Diagnostic Details         File: toopstLcpp:3509 s273_	<	· · · · ·						-					>
Source       Loop Extended       Accomplete Disgnostic Details         File: loopstLcpp:3509 s273       Source       Total Time       %       Loop Time       %         3504       forttime_(stl);	Tan Dama Cause												
Hiel topstl.cpp:3509 5273         Line       Source         Source       Total Time       %       Loop Time       %         3504       forttime_(&tl);	Top Down Source Lo	oop Assembly    7 -	Assistance    ¥ Recommenda	tions 📔 🖬 Co	mpiler Diagn	iostic Detai	115						
Line       Source       Total Time       %       Loop Time       %       A         3504       forttime_(stl);       i_1 = *ntimes;       i_1 = *ntimes;       0.010s         0.200s	File: loopstl.cpp:3509 s27:	3_											
$\frac{3504}{100} = \frac{1}{100} \frac{1}{100}$	Line			Source				To	al Time	%	Loop Time	%	^
3505       i_l = *ntimes;         3506       for (nl = 1; nl <= i_l; ++1)	3504 forttime_(+	«tl);											
3506       for (nl = 1; nl <= i_l; ++nl)	3505 i_1 = *ntin	mes;											
[loop at loopstl.opp:3506 in s273_]         Scalar Loop. Not vectorized: inner loop was already vectorized         No loop transformations were applied         3507         i_2 = *n;         3509         for (i_ = 1; i_ <= i_2; ++i_)	3506 ⊟_for (nl = 1	; nl <= i_1;	++n1)						0.010s l		0.200s	I	
Scalar Loop. Not vectorized: inner loop was already vectorized         No loop transformations were applied         3507         (         3508       i_2 = *n;         3509       for (i_ = 1; i_ <= i_2; ++i_)	[loop at lo	opstl.cpp:350	6 in s273_]										
3507       (         3508       i_2 = *n;         3509       for (i_ = 1; i_ <= i_2; ++i_)	Scalar	Loop. Not vec	torized: inner loop wa	s already '	vectorized								
3507       {         3508       i_2 = *n;         3509       for (i_ = 1; i_ <= 1,2; ++1_)	NO 100p	p cransformaci	ons were applied										
3508       i_2 = *n;         3509       for (i_ = 1; i_ <= i_2; ++i_)	3507 {												
3509       For (1_ = 1; 1_ <= 1_; ++1_)	3508 i_2 = 1	™n;									0.4.50	1	-
Vectorized AVX Loop processing Float32; Float64; Int32 data type(s) having Inserts; Extracts; Masked St Selected (Total Time): 0.010s	509 - for (i_	_ = 1; i <=	1_2; ++1_)						0.010s I		0.160s		
Selected (Total Time): 0.010s	Vectori	ized AVX Loon	<pre>&gt; in 34/3_j nrocessing Float32: Fl</pre>	oat64: Int	32 data tw	me(s) ha	wing Inserts: B	xtracts: Masked St					
	1000011		, , , , , , , , , , , , , , , , , , ,	1/ 1/10	the short of	2-10/ 10		Selected (Total Time):	0.010s				



## Background on loop vectorization

A typical vectorized loop consists of

Main vector body

This is where we want our loops to be executing!

• Fastest among the three!

Optional peel part

• Used for the unaligned references in your loop. Uses Scalar or slower vector

**Remainder part** 

• Due to the number of iterations (trip count) not being divisible by vector length. Uses Scalar or slower vector.

Larger vector register means more iterations in peel/remainder

- Make sure you Align your data!
- Make the number of iterations divisible by the vector length



# Intel Advisor XE shows how much time you are spending in the various parts of your loops!

🚆 Where should I add vectorization and/or threading parallelism? 🗖												
🍄 Summary  😂	urvey	Repor	t 🍅 Refinemer	nt Repo	orts 💧 Annotation	Report 🖞 S	uitability Report					
Elapsed time: 8,52s	Veo	torize	d Not Vector	ized	් FILTER: A	ll Modules	✓ All Source	is V				
Function Call Sites ar	nd Loo	ps		۵	P Vector Issues	Self Time 🕶	Total Time	Loop Туре	Why No Vectoriz	ation?		
🗉 🛄 [loop at fractal.	cpp:17	79 in «	<li>lambda1&gt;::op</li>			0,013sl	12,020s	Collapse	Collapse			
ः 😃 [loop at fract	al.cpp:	179 in	<lambda1>::o</lambda1>	<ul><li>✓</li></ul>	💡 <u>4</u> Serialized use	0,013s1	11,281s 🗔	D Vectorized (Bo	ody)			
i> <sup>(5</sup> ] [loop at fracta	al.cpp:	179 in	<lambda1>::o</lambda1>	-		0,000s l	0,163s1	Peeled				
₃> (] [loop at fracta	al.cpp:	179 in	<lambda1>::o</lambda1>	-		0,000s l	0,576s I	Remainder				
5 [loop at fractal.c	pp:177	7 in <l< td=""><td>ambda1&gt;::oper</td><td></td><td></td><td>0.010s l</td><td>12.030s</td><td>Scalar</td><td></td><td></td><td></td><td></td></l<>	ambda1>::oper			0.010s l	12.030s	Scalar				
<	File: 1	fractal	cpp:164 <lambda1></lambda1>	::opera	tor()							
	Line				Sour	ce			Total Time	%	Loop Time	
	163	Ξ_	fo	or (int	t x = x0; x < x1; +	+x) {					10.822s 🗖	
			[loop at fracta Scalar Loop No loop tra	l.cpp: . Not nsform	163 in <lambdal>::« vectorized: outer ations were applie</lambdal>	operator()] loop was not d	auto-vectorize	d: consider us:				
	164	Ξ		for	(int y = y0; y < y	/1; ++y) {					10.822s	
			[loop at fracta Scalar Loop Loop was un	l.cpp: . Not rolled	164 in <lambdal>::« vectorized: vector by 2</lambdal>	operator()] ization possi	ible but seems	inefficient. U				
	165				fractal_data_array	/[x - x0][y -	y0] = calc_one	e_pixel(x, y, t	10.822s 🗔	)		
	166			}								
	167		}			0		,				
	168		IC	or (111 are:	s y = yu, y_temp =	∪; y < y⊥; + ))•	+y, ++y_temp)	٤				
	170			for	(int x = x0, x ten	up = 0; x < x	1; ++x, ++x ter	) (am				
	171				area.put_pixel(fra	ctal_data_ar	ray[x_temp][y_t	temp]);				
	172			}								
	173		}						0.196s l			



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

	Self	Total			Compiler Vectorizat	
unction Call Sites and Loops.	Time	Time	•	¥	Loop Туре	Why No Vectorization?
[loop in runCForallLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vecto
a[loop in runCForallLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
a[loop in std::basic_string <char,struct std::char_traits<char="">,class std::allo</char,struct>		544.0			Scalar	nonstandard loop is not a vectoriz
[loop in std::basic_string <char,struct std::char_traits<char="">,class std::allo</char,struct>		544.0			Scalar	nonstandard loop is not a vectoriz
[loop in std::num_put <char,class st<="" std::ostreambuf_iterator<char,struct="" td=""><td></td><td>0.234s</td><td></td><td></td><td>Scalar</td><td>nonstandard loop is not a vectoriz</td></char,class>		0.234s			Scalar	nonstandard loop is not a vectoriz

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

#### 2 Issue: Peeled/Remainder loop(s) present

8 All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled /remainder loops to the kernel loop. Read more at <u>Vector Essentials</u>. Utilizing Full Vectors...

Recommendation: Align memory access Projected maximum performance gain: High Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

float \*array;

array = (float \*)\_mm\_malloc(ARRAY\_SIZE\*sizeof(float), 32);

// Somewhere else \_\_assume\_aligned(array, 32); // Use array in loop

### Get Specific Advice For Improving Vectorization Intel® Advisor XE – Vectorization Advisor

Where should I add vectorization and/or threading parallelism? 🖬 🛛 🛛 Intel Advisor XE 2016												
🔗 Summary 🛛 🛠 Survey Report 🔊 Refinemen	nt Repor	rts 💧 Annotation	Report 🛛 🦞 Su	itability Report								
Elapsed time: 8,81s Vectorized Not Vector	ized	් FILTER: A	ll Modules	✓ All Sources	~				Q,			
Function Call Sites and Loops		@ Vector Issues	Self Timew	Total Time	Loon Type	Why No Vectorization?	Vectoriz	ed Loops	^			
	U	Vector issues	Sell Time V	Total Time	соор туре	why no vectorization:	Vecto	Estim	Vector Len	l		
i> <sup>™</sup> [loop at market CIICK TO SEE I	eco	ommenda	ation	11,460s 💳	Scalar					1		
So [loop at arena.cpp:88 in tbb::tbb::]     O 000s1 11,460s Scalar												
🗉 😈 [loop at fractal.cpp:179 in <lambda1>::op</lambda1>		9 5 Ineffective	0,000s	<b>2,022s</b> 0	<u>Collapse</u>	<u>Collapse</u>				l		
i> 🖔 [loop at fractal.cpp:179 in <lambda1>::o</lambda1>		Q 2 Data type co	0,000s I	2,022s 0	Remainder					1		
<									>	l		
			1_		_					i		
Top Down Source Loop Assembly Assistant	ince 🦉	Recommendation	s 📮 Compiler	Diagnostic Deta	ils							
A 2 Issue: Ineffective neeled	l/rom	ainder loon(s	) procont						^	l		
All or some source loop ited	rations	are not executir	in the loop	body Improv	e performan	ce by moving source lo	on iterati	ions fro	m	l		
peeled/remainder loops to	the loo	op body.	ig in the <u>toop</u>	bout improv	e periorina	ice by moving source to	op nerat			l		
Disable unrolling										l		
The trip count after loc	on unro	olling is too smal	II compared to	Advis	orXE	shows hints t	omo		nroll	l		
factor using a directive	2, unit	sing is too sina	n compared o				0 110	Jve	mon	l		
ICL/ICC/ICPC Dire	ctive	IFORT Direction	ve	iterat	ions to	vector body	′ <u>.</u>					
#pragma nounroll		IDIR\$ NOUNRO	L							1		
#pragma unroll IDIR\$ UNROLL												
Read More:												
• User and Reference Guide for the Intel C++ Compiler 15.0 > Compiler Reference > Pragmas > Intel-specific Pragma Reference > unroll/nounroll.												



### Don't Just Vectorize, Vectorize Efficiently

See detailed times for each part of your loops. Is it worth more effort?

🖉 Where should I add vectorization and/or threading parallelism? 🗖												
🄗 Summary 😂 Survey Report 🦻 Refinement Repo	orts 🍐 Annotation	Report 🛛 🆞 Su	iitability Report									
Elapsed time: 8,52s Vectorized Not Vectorized	් FILTER: A	ll Modules	✓ All Sources	~								
Function Call Sites and Loops	P Vector Issues	Self Time -	Total Time	Loop Туре	Why No Vectorization?							
🖃 🖲 [loop at fractal.cpp:179 in <lambda1>::op</lambda1>	♀ <u>4</u> High vector	0,013s1	12,020s 💳	Collapse	Collapse							
🔹 😈 [loop at fractal.cpp:179 in <lambda1>::o 🛛 🗹</lambda1>	💡 🖞 Serialized use	0,013s1	11,281s 🗔	Vectorized (Body)								
i> <sup>™</sup> [loop at fractal.cpp:179 in <lambda1>::o 🗹</lambda1>		0,000s I	0,163s1	Peeled								
i> <sup>™</sup> [loop at fractal.cpp:179 in <lambda1>::o 🗹</lambda1>		0,000s I	0,576s I	Remainder								
🗈 🖱 [loop at fractal.cpp:177 in <lambda1>::oper 🗌</lambda1>		0,010s I	12,030s 📖	Scalar								
<												



### Critical Data Made Easy Loop Trip Counts

Knowing the time spent in a loop is not enough!





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

	Self	Total			Compiler Vectorizat	
iction Call Sites and Loops.	Time	Time	•	¥	Loop Туре	Why No Vectorization?
oop in runCForallLambdaLoops]	0.094s	0.094s			Scalar	vector dependence prevents vector.
oop in runCForallLambdaLoops]	0.140s	3.744s			Scalar	inner loop was already vectorized
oop in stds:basic_string <char,struct stds:char_traits<char="">,class stds:allo</char,struct>		544.0			Scalar	nonstandard loop is not a vectoriza.
oop in std::basic_string <char,struct std::char_traits<char="">,class std::allo</char,struct>		544.0			Scalar	nonstandard loop is not a vectoriza.
oop in std::num_put <char,class st<="" std::ostreambuf_iterator<char,struct="" td=""><td></td><td>0.234s</td><td></td><td></td><td>Scalar</td><td>nonstandard loop is not a vectoriza.</td></char,class>		0.234s			Scalar	nonstandard loop is not a vectoriza.

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

sue: Peeled/Remainder loop(s) present All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled /remainder loops to the kernel loop. Read more at <u>vector issentials</u>.

#### **3. Loop-Carried Dependency Analysis**

Prob	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	٥	Read after write dependency	site2	dqtest2.cpp	dqtest2	Rew New							
	0	Write after write dependency		dqtest2.cpp	dqtest2	R New							
P5	٥	Write after write dependency	site2	dqtest2.cpp	dqtest2	🎙 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							

### Is It Safe to Vectorize?

### Loop-carried dependencies analysis verifies correctness

Program time: 12.82s Vectorized N	lot Vectorized	FILTE	R: All	Module	25	✓ All Source	ces ⊻ Q
Function Call Sites and Loops	Calf Time	Total Time		6	Trip Counts	Compiler Vectoriz	ation
Function Call Sites and Loops	Self Time*	Total Time	œ	¥	Thp Counts	Loop Type	Why No Vectorization?
⊧> <mark>™</mark> [loop at Multiply.c:53 in matvec]	0.047s l	0.047s1			3	Vectorized (Body)	
[loop at Multiply.c:53 in matvec]	0.413s l	0.413s I			101	Scalar	
🗏 🔽 [loop at Multiply.c:45 in matvec]	0.109s l	12.373s 📖		<u> </u>		<u>Collapse</u>	Collapse
🏽 🔽 [loop at Multiply.c:45 in matvec]	0.078s l	11.930s 🔲			12	Vectorized (Body)	
i> [loop at Multiply.c:45 in matvec]	0.031s1	0.444s I			2	Remainder	
> [loop at Driver.c:146 in main]	0.016s	12.483s 🗔	<ul> <li>✓</li> </ul>	<u> 9 1</u>	1000000	Scalar	vector dependence prevents vectoriza
		Calas					
leck Correctness fy and explore loop-carried dependencie: arked loops. <u>Fix</u> the reported problems.	s	Select Co Anal	orre vsis	op ect s an	d		Vector Dependent prevents



## Data Dependencies – Tough Problem #1

Is it safe to force the compiler to vectorize?

Data dependencies

for (i=0;i<N;i++) // Loop carried dependencies!</pre>

A[i] = A[i-1]\*C[i];// Need the ability to check if it

#### // it is safe to force the compiler

#### Issue: Assumed dependency present

The compiler assumed there is an anti-dependency (Write after read - WAR) or true dependency (Read after write - RAW) in the loop. Improve performance by investigating the assumption and handling accordingly.

#### ② Enable vectorization

Potential performance gain: Information not available until Beta Update release

Confidence this recommendation applies to your code: Information not available until Beta Update release

The Correctness 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>.

ICL/ICC/ICPC Directive	IFORT Directive	Outcome
#pragma simd or #pragma omp simd	IDIR\$ SIMD or ISOMP SIMD	Ignores all dependencies in the loop
#pragma ivdep	IDIR\$ IVDEP	Ignores only vector dependencies (which is safest)

Read More:

 User and Reference Guide for the Intel C++ Compiler 15.0 > Compiler Reference > Pragmas > Intel-specific Pragma Reference >

ivdep

omp simd



### Correctness – Is It Safe to Vectorize?

### Loop-carried dependencies analysis

loop_site	_6 main	main.cpp:13	O DAVAGE				
			W RANVET (	🖞 WAR:1 🛛 🛆 WAW:1	91% / 0% / 9%	Mixed strides	
	A 0.0 D	Correction	ss Report		depend	encies	
Memo Problem	y Access Patterns Rep is and Messages	concean					
Memo Problem D	y Access Patterns Rep is and Messages Type	eonteent	iite Name	Sources	Modules State		
Memo Problem ID P1	s and Messages Type Parallel site info	rmation	Site Mame	Sources main.cpp	Modules State test_1.exe  ✓ Not a prol	olem	
Memo Problem D 21 23	s and Messages     Type     Parallel site info     Read after write	rmation I dependence 1	Site Marme ap_site_6 pop_site_6	Sources main.cpp crtexe.c; main.cpp	Modules State test_1.exe ✓ Not a prol test_1.exe 🖡 New	olem	

Wri	te after read dej	pendency: Code L	ocations			
ID	Description	Source	Function	Module	State	
ΞX	17 Read	🖹 main.cpp:22	main	test_1.exe	Re New	
~	20 21	k += a[9]; k *= a[8];				
	22	k -= a[7];				
	23	k += a[6];				
	24	k *= a[5];				
ΞX	18 Read	🖹 main.cpp:23	main		D.A	
	21	k *= a[8];		Course	lines with Dead and	
	22	k -= a[7];		Source	e lines with Read and	
	23	k += a[6];		A . / A.	and the second	
				write a	accesses detected	

Received recommendations to force vectorization of a loop:

1. Mark-up the loop and check for the presence of REAL dependencies

2. Explore dependencies in more details with code snippets

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!

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

	Self	Total			Compiler Vectorizat	
unction Call Sites and Loops.	Time	Time	•	¥	Loop Туре	Why No Vectorization?
[loop in runCForallLambdaLoops]	0.094	s 0.094s			Scalar	vector dependence prevents vector
a[loop in runCForallLambdaLoops]	0.140	s 3.744s			Scalar	inner loop was already vectorized
a[loop in std::basic_string <char,struct std::char_traits<char="">,class std::allo</char,struct>		s 544.0			Scalar	nonstandard loop is not a vectoriza
[loop in std::basic_string <char,struct std::char_traits<char="">,class std::allo</char,struct>		s 544.0			Scalar	nonstandard loop is not a vectoriza
i[loop in std::num_put <char,class st<="" std::ostreambuf_iterator<char,struct="" td=""><td></td><td>s 0.234s</td><td></td><td></td><td>Scalar</td><td>nonstandard loop is not a vectoriza</td></char,class>		s 0.234s			Scalar	nonstandard loop is not a vectoriza

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

#### 2 Issue: Peeled/Remainder loop(s) present

 B
 All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at <u>Vector Essentials</u>. <u>Utilizing Full Vectors</u>...

 S
 Recommendation: Align memory access Projected maximum performance gain: High Projection confidence: Medium

 The compiler created a peeled loop because one of the memory accesses in the source loop does not start at data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

 float \*array;
 array:

 array:
 grapalloc(ARRAY\_SIZE\*sizeof(float), 32);

 // Soewhere else
 see the source loop

#### **3. Loop-Carried Dependency Analysis**

#### 4. Memory Access Patterns Analysis

	۵.	Туре	Site Name	Sources	Modules	State
P1	٢	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a prob
2	۲	Read after write dependency	site2	dqtest2.cpp	dqtest2	Re New
P3	۲	Read after write dependency	site2	dqtest2.cpp	dqtest2	Re New
						R: New
P5	۲	Write after write dependency	site2	dqtest2.cpp	dqtest2	Re New
P6	۲	Write after read dependency	site2	dqtest2.cpp	dqtest2	Re New
P7	۲	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	Re New

Site	e Na	ime	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution		Acces	is Pattern
loop	p_sit	te_203	runCRawLoops	runCRawLoops.coc1063	ORAW:1	No information ava	ilable	No int	formation available
loop	p_sit	te_139	runCRawLoops	runCRawLoops.coc622	No information available	39%/36%/25	6	Mixed	strides
loop	p_sit	te_160	runCRawLoops	runCRawLoops.cxc925	No information available	100%/0%/0%		All un	it strides
M	emo	ory Acc	ess Patterns	orrectness Report					
ID		•	Stride 🕶		Туре	Source	Mod	ules	Alignment
EP.	22		0; 0; 1		Unit stride	runCRawLoops.cxc637	Icals.	exe	
2	63	5		j2 = ( j2 & 64-1	) ;	56			
	63	86		p[ip][0] += y[i2+	32];				
	63	37		p[ip][1] += z[j2+	32];				
	63	88		i2 += e[i2+32];					
	63	9		j2 += f[j2+32];					
œ P.	23		0;0		Unit stride	runCRawLoops.cxc638	Icals.	exe	
BP:	30		-1575; -63; -26; -3	25; -1; 0; 1; 25; 26; 63; 21648	01 Variable stride	runCRawLoops.cxc628	Icals.	exe	
2	62	26		il 4= 64-1;					
	62	27		j1 4= 64-1;					
	62	28		p[ip][2] += b[j1]	[11];				

## Non-Contiguous Memory – Tough Problem #2

Potential to vectorize but may be inefficient

Non-unit strided access to arrays

```
for (i=0;i<N;i+=2) //Incrementing "i" by 2 is not unit stride</pre>
```

//We need a way to check how we are

//accessing memory.

Indirect reference in a loop

```
for (i=0;i<N;i++)</pre>
```



# Improve Vectorization

### Memory Access pattern analysis

🖉 Where should I add vectorization and/or threading parallelism? 🗖												
🍄 Summary 😂 Survey Report 🍅 Refinement Reports 💧 Annotation Report 🖞 Suitability Report												
Elapsed time: 8,52s Vectorized Not Vectorized	FILTER: All	Modules 🗸 🗸	All Sources	~								
Function Call Sites and Loops	Select loop	os of inter	est	Loop Туре	Why No Vectorization?							
🖃 💹 [loop at fractal.cpp:179 in <lambda1>::op</lambda1>	🧟 🖌gn vector	0,013s1	12,020s 🥅	<u>Collapse</u>	<u>Collapse</u>							
👔 😈 [loop at fractal.cpp:179 in <lambda1>::o 🛛 🗹</lambda1>	9 4 Serialized use	0,013s1	11,281s 📖	Vectorized (Body)								
i> <sup>™</sup> [loop at fractal.cpp:179 in <lambda1>::o 🗹</lambda1>		0,000s I	0,163s l	Peeled								
i> <sup>™</sup> [loop at fractal.cpp:179 in <lambda1>::o</lambda1>		0,000s I	0,576s I	Remainder								
i> 🗗 [loop at fractal.cpp:177 in <lambda1>::oper</lambda1>		0,010s l	12,030s 📖	Scalar								
<												





### Find vector optimization opportunities Memory Access pattern analysis

Check memory access patterns in your application 🌳 Summary 🛛 😽 Survey Report 🏠 Refinement Reports 🛛 🛕 Annotation Report 🔍 Suitability Report Site Function Site Info Strides Distribution Access Pattern Site Name Loop-Carried Dependencies 100% / < 1,0000% / ... Mixed st loop site 79 operator() fractal.cpp:179 No information available oop site 93 | operator() fractal.cpp:179 No information available 100% / 0% / 0% All unit strides loop\_site\_94 operator() fractal.cpp:179 No information available All memory accesses are uniform, with zero unit stride, so the same data is read in each iteration Memory Access Patterns Report We can therefore declare this function using the omp • Stride Type syntax: pragma omp declare simd uniform(x0 44 0 ∃P21 44 0 Unit stride fractal.cpp:66 fractal.exe 64 color\_t color; 65 66 fx0 = x0 - size x / 2.0f; 67 fy0 = y0 - size y / 2.0f; 68 fx0 = fx0 / magn + cx;= P24 🛛 0 Unit stride fractal.cpp:68 fractal.exe 66 fx0 = x0 - size x / 2.0f;67 fy0 = y0 - size y / 2.0f; 68 fx0 = fx0 / magn + cx;69 fv0 = fv0 / magn + cv;4-4 0 Unit stride fractal.cpp:69 fractal.exe 0 + P30 Unit stride fractal.cpp:74 fractal.exe



Stride distribution

# Quickly Find Loops with Non-optimal Stride

Memory Access pattern analysis

- Quickly identify loops that are good, bad or mixed.
- Unit stride memory accesses are preferable.
- Find unaligned data

« M	🛚 🙋 Check memory access patterns in your application 🖿 👘 👘 👘 Intel Advisor XE 2010											
ዋ Su	mmary	😂 Sur	vey Report 🍅 Refinem	nent Reports 🛛 🚡 MAP Source:	fractal.cpp	🍐 Annota	tion Report 🛛 🦞 Suitability Rep	oort				
Site Na	ame	Site Fun	ction Site Info	Loop-Carried Dependencies	Strides Distrib	oution	Access Pattern					
loop_si	ite_54	operator	r() fractal.cpp:164	ON dependencies found	No informatio	on available	No information available					
loop_si	ite_129	operator	r() fractal.cpp:164	No information available	100% / 0	%/0%	All unit strides					
Memo	orv Acc	ess Patter	ns Report Correctness	Report								
ID	•	Stride	Type	Source	Modules	Alignment						
±P1	i		Parallel site information	n fractal.cpp:164	fractal.exe	-						
<b>⊟ P</b> 3	<u>83</u>	0	Unit stride	fractal.cpp:100	fractal.exe							
98	B	}										
10	9 #e: 00	int b	= (int)(256 * mu);									
10	01	int g	= (b / 8);									
10	02	int r	= (g / 16);									
<b>□ P4</b>		0	Unit stride	fractal.cpp:164	fractal.exe							
10	62											
16	63 ca		for (int x =	= x0; x < x1; ++x) {								
16	65		frac	ctal data arravíx - x0)(v	/ - v01 = ca	lc one pi	xel(x, v, tmp max iterat	tions, tmp size x, tmp si				
16	66		}					,,				
± P5		0	Unit stride	fractal.cpp:164	fractal.exe							
<b>⊞ P6</b>	<u>83</u>	0; 1	Unit stride	fractal.cpp:165	fractal.exe							
± P7		0; 1	Unit stride	fractal.cpp:165	fractal.exe							
<b>± P8</b>		0	Unit stride	fractal.cpp:60	fractal.exe							
		0	11.20.001	1	e							



# **GETTING STARTED**

## Before you analyze

**Create Project** 

 File->New->Project vecsampple - Project Properties Analysis Target Binary/Symbol Search Source Search Survey A Target type: Survey/Suitability Launch Application V Survey/Suitability Launch Application - To Surve 😑 🗁 Sufvey:Trip Count Analysis figure the application executable (target) to analyze. Press F1 for more details. 😽 Mem No application executable (target) file specified. 👘 Corre Application: Browse... ¥ Application parameters: Modify... directory as working directory ? Create a Project Browse... ~ onment variables: Project name: vecsampple Modify... filing mode: Auto V C:\advisor\_samples\vec\_samples Location: Browse... Create Project Cancel < > OK Cancel



# Analyze what loops you are spending your time in and how they have been vectorized!

<ol> <li>Survey Ta Explore wher and/or threa</li> <li>Collect</li> <li>Command L</li> </ol>	arget re to add e ding.	ficient	vectorization			Click	Collect						
1.1 Find Trip Find how ma ▶ Collect Command L	Counts any iteration	ns are e	xecuted.				Survey	Rep	ort				
Mark Loops Select loops	for Deep	er Analı	/sis										15 224 6
Correctness Patterns and	🖉 wn	ere sr	Suprey Report	ectorizat	ion and/or	A Appositation	Parallelism:	ort			Intel Ac	lvisor 2	XE 2016
There are	Flansed t	ime 15	47s Usctor	ized (5 Not)	Vectorized	EIL TER-		Join		Sources V			Q
	clopsed t	inici i s	Vector		Vectorized		Air Wouldes	Vectoriz	red Loop	sources ·	4	Instruct	tion Set Analysi
2.1 Check (	Loops	۵	Vector Issues	Self Time▼	Total Time	Loop Type	Why No Vectorization?	Vecto	Estim	Vector Length	Compiler Estimated Gain	Traits	Data Types
dependenci	iə 🖱 [lo		§ 1 Assumed	. 14.030s 🗖	14.030s 🚍	Scalar	vector dependence						Float64
reported pro	i> 🖱 [lo			0.985s I	15.015s 🥅	Scalar	uter loop was not a						Float64
Collect	i> 🖱 [lo			0.000s l	15.035s 💳	Scalar	Ioop with function c						Float64
Command I C.2 Check N Identify and accesses for problems. Collect	lemory Ad explore co marked lo	: <b>cess</b> P mplex r ops. Fix	atterns nemory the reported										



### All the data in one place





# Next analyze how many times your loops are iterating and how many times they are called.

Explore where to add efficient vectorization mod/or threading. Command Line 1.1 Find Trip Counts Find how many iterations are executed. Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Collect Coll	1. Survey Target			:K							
Collect       Image: Command Line         L1 Find Trip Counts       Mark Loops for Deeper Analysis         Find how many iterations are executed.       Image: Command Line         Collect       Image: Command Line         Mark Loops for Deeper Analysis         Select loops in the Survey result for         Correct parter       Image: Command Line         Mark Loops for Deeper Analysis         Select loops in the Survey result for         Correct parter       Image: Command Line         Vector Issues       Self Time marked loops in the Survey result for         Correct parter       Image: Command Line         Vector Issues       Self Time marked loops in the Survey result for         Correct parter       Image: Command Line       Image: Command Line         Vector Issues       Self Time marked loops in the Survey result for       Image: Command Line         V © [loop at Multiply.cc55 in matyce]       Image: Parter       Image: Command Line         V © [loop at Multiply.cc44 in matyce]       Image: Command Line       Image: Command Line         Collect [marked loops: Fix the reported problems.       Image: Collect Image: Coll	xplore where to add efficient vectorization and/or threading.		Coll	ec	Trip Counts	s				×	1
Command Line       50       50       50       101000000       < 0.0001s         1.1 Find Trip Counts       Find how many iterations are executed.       101       101       101       1000000       1       < 0.0001s         Command Line       Command Line       Command Line       Image: Command Line       Image: Command Line       Image: Command Line       Total Time       Trip Counts       Command Line         Mark Loops for Deeper Analysis       Concect       Image: Command Line       Total Time       Trip Counts       Collect:       Command Line         Image: Command Line       Image: Command Line       Image: Command Line       Total Time       Total Time       Trip Counts       Collect:	Collect		t		Median	Min	Max	Call Co	ount	Iteration Duration	
1.1 Find Trip Counts         Find how many iterations are executed.         Collect         Collect         Command Lins             Mark Loops for Deeper Analysis    Select loops in the Survey result for Correc Patter Loops              10       101       101       1000000       1000000       1       < 0.0001s    Select loops in the Survey result for Correc Patter Loops              10       101       101       101       101       1000000       1       < 0.0001s       Call Count       Collect       Collect       Collect       Median       Min       Max       Call Count       Iteration Duration       Call Count       Loop                10       101       101       101       101       101000000       < 0.0001s       Sca         2.1 Ch       10       [loop at Multiply.cc:44 in mativec]       0.98551       15.0155       101       101       101       10000000       1       < 0.0001s       Sca         2.1 Ch       10       [loop at Driver.c:145 in main]       0.0000s1       15.035s       10000000       1       < 0.0001s       Sca         Corect Problems.	Command Line				50	50	50	10100	0000	< 0.0001s	
ind how many iterations are executed.     Collect     Collect <td>I.1 Find Trip Counts</td> <td></td> <td></td> <td></td> <td>101</td> <td>101</td> <td>101</td> <td>100000</td> <td>0 .</td> <td>&lt; 0.0001s</td> <td>1</td>	I.1 Find Trip Counts				101	101	101	100000	0 .	< 0.0001s	1
Collect Command Line  Aark Loops for Deeper Analysis elect loops in the Survey result for Correct Total Time Total Time Total Time Total Time Total Time Median Min Max Call Count Iteration Duration Loc So Content So Con	ind how many iterations are executed.				1000000	1000000	1000000	1		< 0.0001s	
Aark Loops for Deeper Analysis elect loops in the Survey result for Total Time Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loops Loop	Collect										
attern The Loops       Vector Issues       Self Time +       Total Time +       Total Time +       Total Time +       Init Events       Call Count       Iteration Duration       Loops         ab * © [loop at Multiply.c:55 in matvec]       Image: Call Count       Image: Call Count       Iteration Duration       Call Count       Call Count       Iteration Duration       Call Count       Iteratio	ark Loops for Deeper Analysis elect loops in the Survey result for orrec					Trin Court				L. L	21
Solution of the second sec	atterr Loops	۵	Vector Issues	Self Time <del>▼</del>	Total Time	Median	Min	Max	Call Cour	nt Iteration Duration	Loop
1 Ch       io C [loop at Multiply.c:44 in matvec]       0.985s1       15.015s       101       101       1000000       < 0.0001s       Sca         io C [loop at Driver.c:145 in main]       0.000s1       15.035s       1000000       1000000       1       < 0.0001s       Sca         ependencies for marked loops. Fix the ponted problems.       0.000s1       15.035s       1000000       1000000       1       < 0.0001s       Sca         Collect       Image: Collect collect       Image: Collect collec	i> <sup>™</sup> [loop at Multiply.c:55 in matvec]		§ 1 Assumed de	14.030s	14.030s 💳	50	50	50	1010000	000 < 0.0001s	Scala
Image: Control of the product of th	1 cb i> <sup>(5</sup> [loop at Multiply.c:44 in matvec]			0.985s I	15.015s 📖	0 101	101	101	1000000	< 0.0001s	Scalar
ependencies for marked loops. <u>Fix</u> the eported problems. Collect Command Line - Nothing to analyze .2 Check Memory Access Patterns lentify and explore complex memory	tentif i> 🖉 [loop at Driver.c:145 in main]			0.000s I	15.035s	1000000	1000000	1000000	1	< 0.0001s	Scala
roblems.											
Command Line	Command Line										



### Specify loops for deeper analysis

🖉 Where should I add vectorization and/or threading parallelism? 🗖											
🍄 Summary 💐 Survey Report 🦻 Refinement Reports 💧 Annotation Report 🦞 Suitability Report											
Elapsed time: 15.47s 🕑 Vectorized 💿 Not Vectorized 💿 FILTER: All Modules 🗸 All Sources 🗸											
Loops	۵	Vector Issues	Self Time 🔻	Total Time	Loop Туре	Why No Vectorization?					
₃> 🖱 [loop at Multiply.c:55 in matvec]	~		14.030s 🗖	14.030s 📼	Scalar	vector dependence p					
i> 🖔 [loop at Multiply.c:44 in matvec]	✓		0.985s I	15.015s 💶	Scalar	outer loop was not a					
း 🕑 [loop at Driver.c:145 in main]	✓		0.000s	15.035s 🗔	Scalar	Ioop with function c					



### Deeper analysis

Command Line

Command Line

#### Check dependencies





Identify and explore complex memory accesses for marked loops. Fix the reported problems.





### **Deeper analysis**

#### Memory Access Pattern analysis

I. Survey Target
explore where to add efficient vectorization and/or threading.
Collect
Command Line

1.1 Find Trip Counts Find how many iterations are executed. Collect <u>Command Line</u>

Mark Loops for Deeper Analysis Select loops in the Survey result for Correctness and/or Memory Access Patterns analysis. -- There are NO marked loops --

2.1 Check Correctness

Identify and explore loop-carried dependencies for marked loops. <u>Fix</u> the reported problems.

Collect

🍐 -- Nothing to analyze --

#### 2.2 Check Memory Access Patterns

Identify and explore complex memory accesses for marked loops. Fix the reported problems.

🕨 Collect 🖻

Command Line

🖉 Check memory access patterns in your appli 👘 n 🗖											
🤗 Summary 🛛 😂 Survey Repo	ort 🍅 Refinement Reports	🍐 An	ation Report	🖞 Suitability Report							
Site Location	Loop-Carried Dependencies	Stride	Distribution	Access Pattern	Site Name						
loop at Driver.c:145 in main	No dependencies found	100	% / 0% / 0%	All unit strides	loop_site_6						
loop at Multiply.c:44 in matvec	No dependencies found	859	6 / 15% / 0%	Mixed strides	loop_site_10						
loop at Multiply.c:55 in matvec	RAW:1	749	6 / 26% / <mark>0%</mark>	Mixed strides	loop_site_8						

Stride distribution

	Memory Access Patterns Report Correctness Report													
	ID	•	Stride	Туре	Source	Nested Function	Modules	Alignment						
	<b>⊞</b> P3	i		Parallel site information	Driver.c:145		matrix_vector_multiplication_c.exe							
	<b>⊞ P9</b>	14	0	Unit stride	Driver.c:157		matrix_vector_multiplication_c.exe							
1	± P10	1	0	Unit stride	Multiply.c:39	matvec	matrix_vector_multiplication_c.exe							
I	± P12	44	0	Unit stride	Multiply.c:44	matvec	matrix_vector_multiplication_c.exe							
I	⊟ <b>P1</b> 4	•••	0; 1	Unit stride	Multiply.c:45	matvec	matrix_vector_multiplication_c.exe							
I	4	3	int i	, j;										
I	4	4												
I	4	5	<pre>for (i = 0; i &lt; size1; i++) {</pre>											
I	4	6	b[i] = 0;											
I	4	7												

Click Collect



### **Configurations for Binomial Options SP**



#### Platform Hardware and Software Configuration

#### **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

Performance measured in Intel Labs by Intel employees

	Unscaled																
	Core			L1					Memory		H/W						
	Frequenc	Cores/	Num	Data	L1 I	L2	L3		Frequenc	Memory	Prefetchers	HT	Turbo		O/S	Operating	Compiler
Platform	У	Socket	Sockets	Cache	Cache	Cache	Cache	Memory	у	Access	Enabled	Enabled	Enabled	C States	Name	System	Version
Intel® Xeon™														Disable	Fedora	3.11.10-	icc version
5472 Processor	3.0 GHZ	4	2	32K	32K	12 MB	None	32 GB	800 MHZ	UMA	Y	Ν	Ν	d	20	301.fc20	14.0.1
Intel® Xeon™									1333					Disable	Fedora	3.11.10-	icc version
X5570 Processor	2.93 GHZ	4	2	32K	32K	256K	8 MB	48 GB	MHZ	NUMA	Y	Y	Y	d	20	301.fc20	14.0.1
Intel® Xeon™									1333					Disable	Fedora	3.11.10-	icc version
X5680 Processor	3.33 GHZ	6	2	32K	32K	256K	12 MB	48 MB	MHZ	NUMA	Y	Y	Y	d	20	301.fc20	14.0.1
Intel® Xeon™ E5									1600					Disable	Fedora	3.11.10-	icc version
2690 Processor	2.9 GHZ	8	2	32K	32K	256K	20 MB	64 GB	MHZ	NUMA	Y	Y	Y	d	20	301.fc20	14.0.1
Intel® Xeon™ E5																	
2697v2									1867					Disable	Fedora	3.11.10-	icc version
Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	MHZ	NUMA	Y	Y	Y	d	20	301.fc20	14.0.1
Codename									2133					Disable	Fedora	3.13.5-	icc version
Haswell	2.2 GHz	14	2	32K	32K	256K	35 MB	64 GB	MHZ	NUMA	Y	Y	Y	d	20	202.fc20	14.0.1



# 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 © 2015v, 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

