VECTORIZE OR DIE TUTORIAL

Legal Notice

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

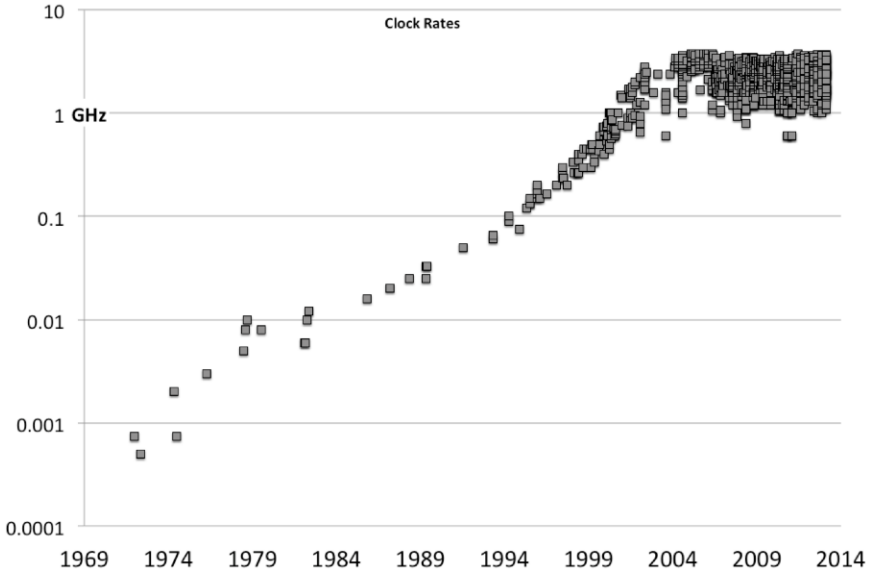
New possible treatments for Parkinson's

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

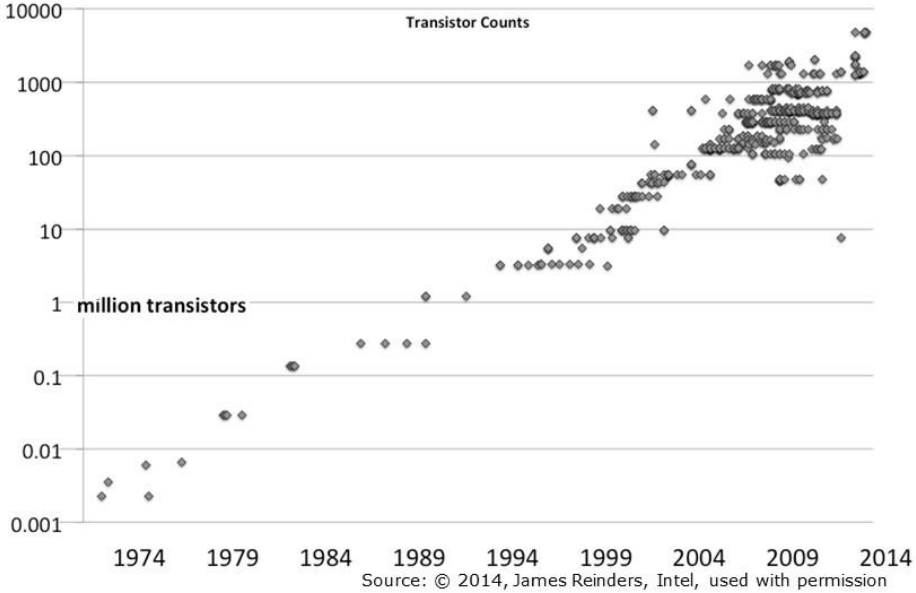


Source: © 2014, James Reinders, Intel, used with permission

Software must be parallelized to realize 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 → More Threads → Wider vectors

	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge EP	Intel® Xeon® processor code-named Ivy Bridge EP	Intel® Xeon® processor code-named Haswell EP	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing[1]
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

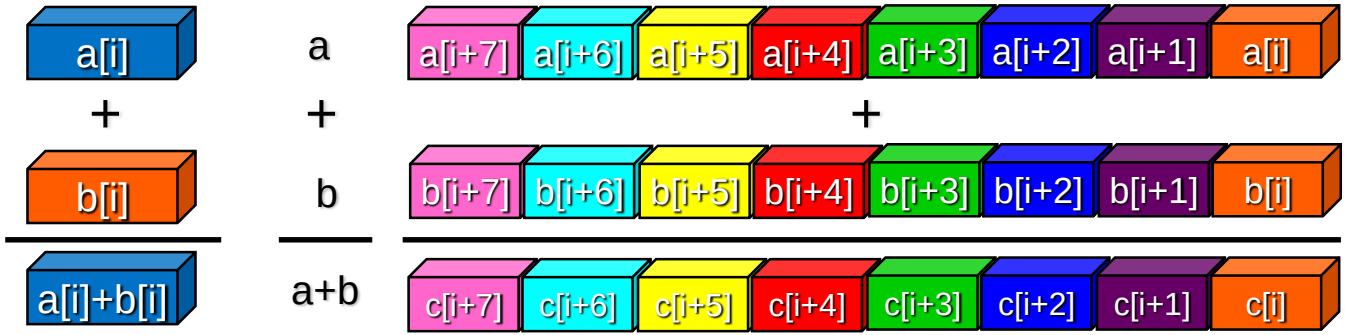
VECTORIZATION OPTIMIZATION AND THREAD PROTOTYPING

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];
```



Vector Instructions are Dramatically Faster

Multiple arithmetic operations with a single instruction

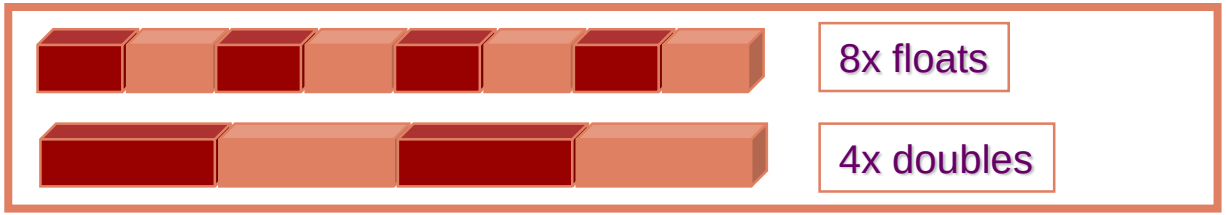
Adding 2 vectors

	4.4	1.1	3.1	-8.5	-1.3	1.7	7.5	5.6	-3.2	3.6	4.8
+	-0.3	-0.5	0.5	0	0.1	0.8	0.9	0.7	1	0.6	-0.5
=	4.1	0.6	3.6	-8.5	-1.2	2.5	8.4	6.3	-2.2	4.2	4.3

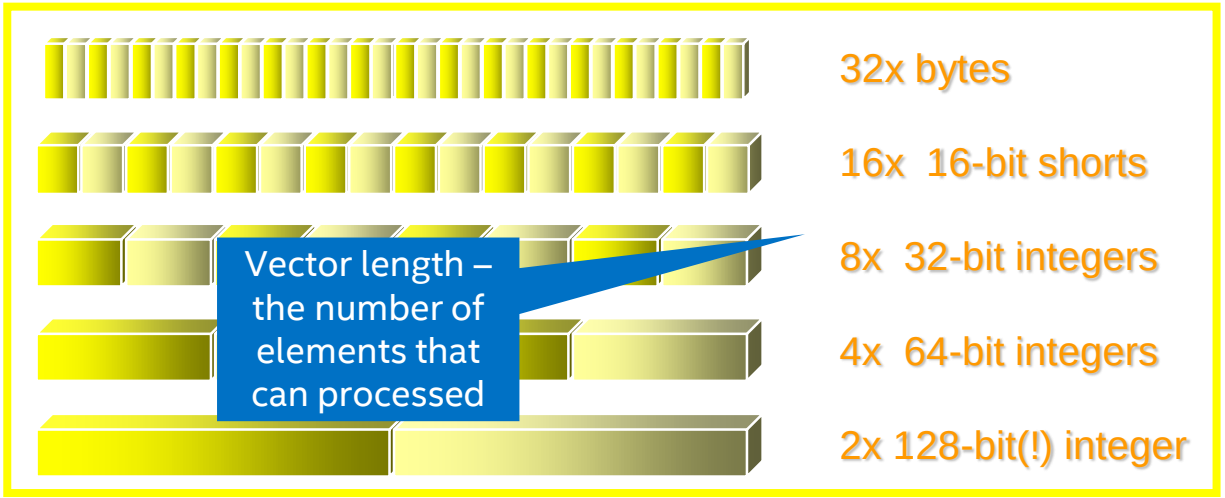
- These instructions are also referred to as Single Instruction Multiple Data (SIMD instructions)

Intel® Advanced Vector Extensions (Intel® AVX)

Intel® AVX



Intel® AVX2



Don’t use a single Vector lane!

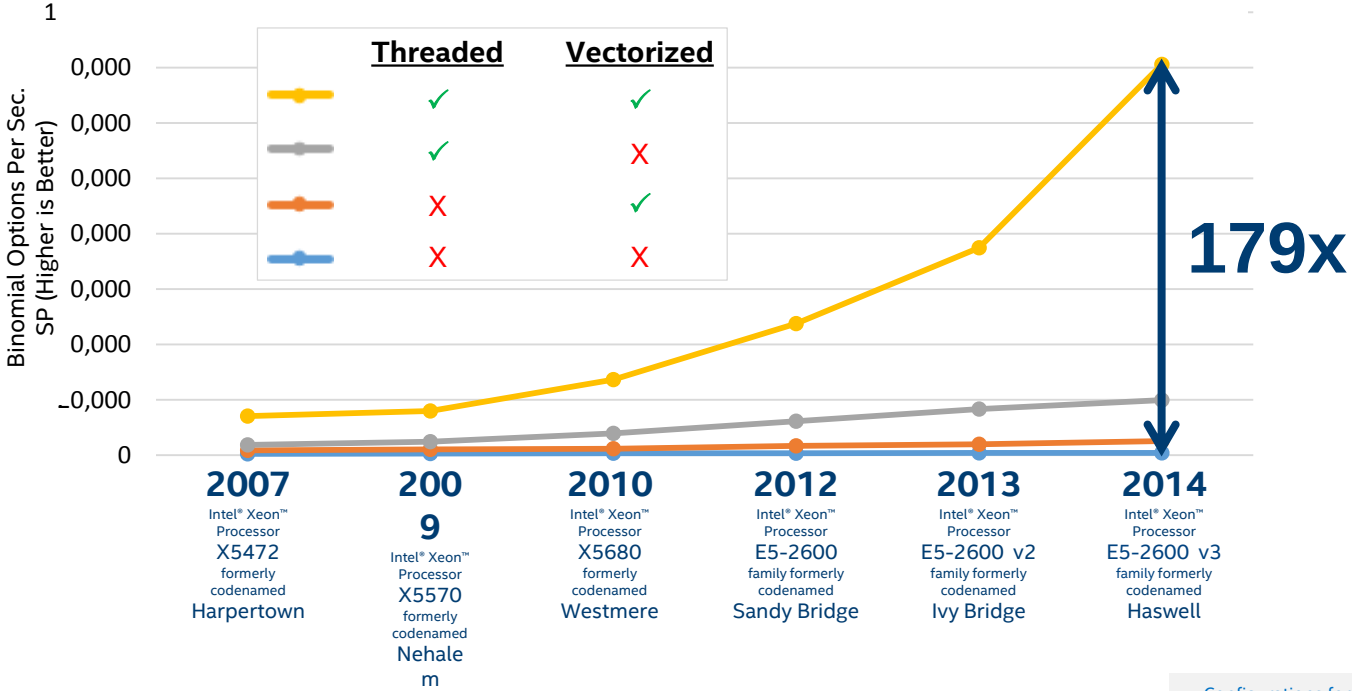
Un-vectorized and un-threaded software will under perform

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 http://www.intel.com/performance

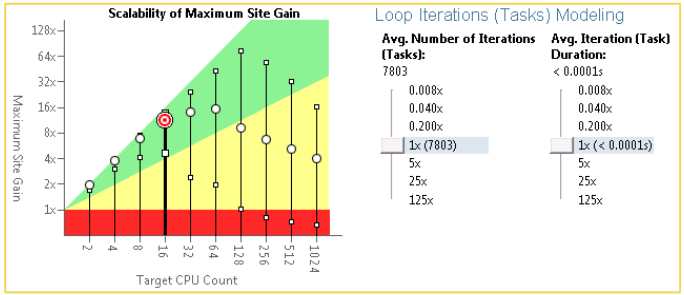
Configurations for Binomial Options SP at the end of this presentation

Data-Driven Threading Design

Intel® 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

http://intel.ly/advisor-xe

Data Driven Vectorization Design

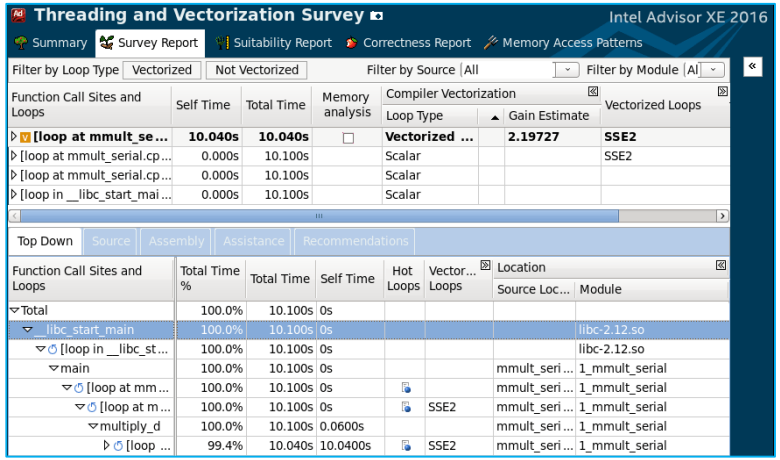
Intel® Advisor XE – Vectorization Advisor

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

The Right Data At Your Fingertips

Get all the data you need for high impact vectorization

Filter by which loops are vectorized!

Trip Counts

What prevents vectorization?

Where sh dd vectorization and/or threading parallelism?

Intel Advisor XE 2016

Summary | S Report | Refinement Reports | Annotation Report | Suitability port

Elapsed time: 54.44s | ctorized | Not Vectorized | FILTER: All Modules | l Sources

Function Call Sites and Loops		Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops		
								Vecto...	Efficiency	Vector L...
[loop at stl_algo.h:4740 in std::tr...			0.170s	0.170s		Scalar	non-vectorizable loop ins...			
[loop at lo pstl.cpp:2449 in s234_]		2 Ineffective peeled/rem...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX	~100%	4
[loop a loopstl.cpp:2449 in s...			0.150s	0.150s	12	Vectorized (Body)		AVX		4
[loop loopstl.cpp:2449 in s...			0.020s	0.020s	4	Remainder				
[loop at pstl.cpp:7900 in vas_]			0.170s	0.170s	500	Scalar	vectorization possible but...			4
[loop a opstl.cpp:3509 in s2...		**1 High vector register ...**	**0.160s**	**0.160s**	**12**	**Expand**	**Expand**	**AVX**	**~69%**	**8**
[loop opstl.cpp:3891 in s279_]		2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~96%	8
[loop opstl.cpp:6249 in s414_]			0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop l_numeric.h:247 in std...		1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve...			

Focus on hot loops

What vectorization issues do I have?

Which Vector instructions are being use?

How efficient is the code?

4 Steps to Efficient Vectorization

Intel® Advisor XE – Vectorization Advisor

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization: Loop Type	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
[loop in std::_Complex_base<double,struct _C_double_complex>::si...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	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

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 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

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	New
P7	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	New

4. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cxx:1063	RAW:1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cxx:622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cxx:925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns | Correctness Report

ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cxx:637	lcals.exe	
P23	0; 0	Unit stride	runCRawLoops.cxx:638	lcals.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cxx:628	lcals.exe	

```
635        j2 = ( j2 & 64-1 ) ;
636        p[ip][0] += y[i2+32];
637        p[ip][1] += z[j2+32];
638        i2 += e[i2+32];
639        j2 += f[j2+32];
```

```
626        i1 &= 64-1;
627        j1 &= 64-1;
628        p[ip][2] += b[j1][i1];
```

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time			Compiler Vectorization	
					Loop Type	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
[loop in std::_Complex_base<double,struct _C_double_complex>::i...	0.031s	0.031s			Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stmts were reordered						
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...			Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s			Scalar	nonstandard loop is not a vectoriza...

Efficiently Vectorize your code

Intel Advisor XE – Vectorization Advisor

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary | Survey Report | Refinement Reports | Annotation Report | Suitability Report

Elapsed time: 54.44s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops: Vecto...	Efficiency	Vector L...
[loop at stl_algo.h:4740 in std::tr...		0.170s	0.170s		Scalar	non-vectorizable loop ins...			
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX	~100%	4
[loop at loopstl.cpp:2449 in s...		0.150s	0.150s	12	Vectorized (Body)		AVX		4
[loop at loopstl.cpp:2449 in s...		0.020s	0.020s	4	Remainder				
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but...			4
[loop at loopstl.cpp:3509 in s2...	**1 High vector register ...**	**0.160s**	**0.160s**	**12**	**Expand**	**Expand**	**AVX**	**~69%**	**8**
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~96%	8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop at stl_numeric.h:247 in std...	1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve...			

Top Down | Source | Loop Assembly | Assistance | Recommendations | Compiler Diagnostic Details

File: loopstl.cpp:3509 s273_

Line	Source	Total Time	%	Loop Time	%
3504	forttime_ (&tl);				
3505	i__1 = *ntimes;				
3506	for (nl = 1; nl <= i__1; ++nl)	0.010s		0.200s	
	[loop at loopstl.cpp:3506 in s273_] 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__)	0.010s		0.160s	
	[loop at loopstl.cpp:3509 in s273_] Vectorized AVX Loop processing Float32; Float64; Int32 data type(s) having Inserts; Extracts; Masked St				
	Selected (Total Time):	0.010s			

Background on loop vectorization

A typical vectorized loop consists of

Main vector body

- 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!

This is where we want our loops to be executing!

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 | 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		Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op ...		4 High vector ...	0,013s	12,020s	Collapse	Collapse
[loop at fractal.cpp:179 in <lambda1>::o ...	✓	4 Serialized use ...	0,013s	11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o ...	✓	2 Data type co ...	0,000s	0,163s	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o ...	✓	2 Data type co ...	0,000s	0,576s	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper ...		2 Data type co ...	0,010s	12,030s	Scalar	

File: fractal.cpp:164 <lambda1>::operator()

Line	Source	Total Time	%	Loop Time	%
163	for (int x = x0; x < x1; ++x) {			10.822s	
	[loop at fractal.cpp:163 in <lambda1>::operator()] Scalar Loop. Not vectorized: outer loop was not auto-vectorized: consider us: No loop transformations were applied				
164	for (int y = y0; y < y1; ++y) {			10.822s	
	[loop at fractal.cpp:164 in <lambda1>::operator()] Scalar Loop. Not vectorized: vectorization possible but seems inefficient. U: Loop was unrolled by 2				
165	fractal_data_array[x - x0][y - y0] = calc_one_pixel(x, y, t	10.822s			
166	}				
167	}				
168	for (int y = y0, y_temp = 0; y < y1; ++y, ++y_temp) {				
169	area.set_pos(0, y - y0);				
170	for (int x = x0, x_temp = 0; x < x1; ++x, ++x_temp) {				
171	area.put_pixel(fractal_data_array[x_temp][y_temp]);				
172	}				
173	}	0.196s			

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization: Loop Type	Compiler Vectorization: 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
[loop in std::_Complex_base<double,struct _C_double_complex>::i...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo ...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza ...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo ...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza ...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st ...	0.000s	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

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 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
```

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 | Refinement Reports | Annotation Report | Suitability Report

Elapsed time: 8,81s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops		
						Vecto...	Estim...	Vector Len
[loop at market			11,460s	Scalar				
[loop at arena.cpp:88 in tbb::tbb::]		0,000s	11,460s	Scalar				
[loop at fractal.cpp:179 in <lambda1>::op ...	5 Ineffective ...	0,000s	2,022s	Collapse	Collapse			
[loop at fractal.cpp:179 in <lambda1>::o ...	2 Data type co ...	0,000s	2,022s	Remainder				

Click to see recommendation

Top Down | Source | Loop Assembly | Assistance | Recommendations | Compiler Diagnostic Details

3 | 5

Issue: Ineffective peeled/remainder loop(s) present

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

Disable unrolling

The trip count after loop unrolling is too small compared to ... nroll factor using a directive.

ICL/ICC/ICPC Directive	IFORT Directive
#pragma nounroll	!DIR$ NOUNROLL
#pragma unroll	!DIR$ UNROLL

Read More:

- User and Reference Guide for the Intel C++ Compiler 15.0 > **Compiler Reference > Pragmas > Intel-specific Pragma Reference > unroll/nounroll.**

Advisor XE shows hints to move iterations to vector body.

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 Reports | Annotation Report | Suitability Report

Elapsed time: 8,52s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops		Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op ...		**4 High vector ...**	**0,013s**	**12,020s**	**Collapse**	**Collapse**
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	4 Serialized use ...	0,013s	11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	2 Data type co ...	0,000s	0,163s	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	2 Data type co ...	0,000s	0,576s	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper ...	☐	2 Data type co ...	0,010s	12,030s	Scalar	

Critical Data Made Easy

Loop Trip Counts

Knowing the time spent in a loop is not enough!

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary | Survey Report | Refinement Reports | Annotation Report | Suitabilit

Program time: 12.82s | Vectorized | Not Vectorized | FILTER: All Modul | All Sources

Function Call Sites and Loops	Self Time	Total Time			Trip Counts				Compiler Vectorization	
					Median	Min	Max	Call Count	Loop Type	Why No Vectorizatio
[loop at Multiply.c:53 in matvec]	11.898s	11.898s		1					Collapse	Collapse
[loop at Multiply.c:53 in matvec]	11.851s	11.851s		1	101	101	101	12000000	Vectorized (Body)	vector dependence p
[loop at Multiply.c:53 in matvec]	0.047s	0.047s			3	3	3	1000000	Vectorized (Body)	
[loop at Multiply.c:53 in matvec]	0.413s	0.413s			101	101	101	2000000	Scalar	
[loop at Multiply.c:45 in matvec]	0.109s	12.373s		1					Expand	Expand
[loop at Driver.c:146 in main]	0.016s	12.483s		1	1000000	1000000	1000000	1	Scalar	vector dependence p

Check actual trip counts

Loop is iterating 101 times but called > million times

Since the loop is called so many times it would be a big win if we can get it to vectorize.

1.1 Find Trip Counts

Find how many iterati are executed.

Command Line

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization: Loop Type	Compiler Vectorization: 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
[loop in std::_Complex_base<double,struct _C_double_complex>::...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	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

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 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

Problems and Messages

ID		Type	Site Name	Sources	Modules	State
P1		Parallel site information	site2	dqtest2.cpp	dqtest2	✔ Not a problem
P2		Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P3		Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P4		Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P5		Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P6		Write after read dependency	site2	dqtest2.cpp	dqtest2	New
P7		Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	New

Is It Safe to Vectorize?

Loop-carried dependencies analysis verifies correctness

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary | Survey Report | Refinement Reports | Annotation Report | Suitability Report

Program time: 12.82s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops	Self Time	Total Time			Trip Counts	Compiler Vectorization	
						Loop Type	Why No Vectorization?
[loop at Multiply.c:53 in matvec]	0.047s	0.047s	☐		3	Vectorized (Body)	
[loop at Multiply.c:53 in matvec]	0.413s	0.413s	☐		101	Scalar	
[loop at Multiply.c:45 in matvec]	0.109s	12.373s		1		Collapse	Collapse
[loop at Multiply.c:45 in matvec]	0.078s	11.930s	☐		12	Vectorized (Body)	
[loop at Multiply.c:45 in matvec]	0.031s	0.444s	☐		2	Remainder	
[loop at Driver.c:146 in main]	0.016s	12.483s	☑	1	1000000	Scalar	vector dependence prevents vectoriza...

Vector Dependence prevents Vectorization!

Select loop for Correct Analysis and press play!

2.1 Check Correctness

Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.

Command Line

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!

    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.

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 directive.

ICL/ICC/ICPC Directive	IFORT Directive	Outcome
#pragma simd or #pragma omp simd	!DIR$ SIMD or !$OMP SIMD	Ignores all dependencies in the loop
#pragma ivdep	!DIR$ 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

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 WAR:1 WAW:1	91% / 0% / 9%	Mixed strides

Detected dependencies

Memory Access Patterns Report | Correctness Report

Problems and Messages

ID		Type	Site Name	Sources	Modules	State
P1		Parallel site information	loop_site_6	main.cpp	test_1.exe	Not a problem
P3		Read after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New
P4		Write after write dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New
P5		Write after read dependency	loop_site_6	crtexe.c; main.cpp	test_1.exe	New

Write after read dependency: Code Locations

ID	Description	Source	Function	Module	State
X17	Read	main.cpp:22	main	test_1.exe	New

```
20  k += a[9];
21  k *= a[8];
22  k -= a[7];
23  k += a[6];
24  k *= a[5];
```

X18 Read main.cpp:23 main

```
21  k *= a[8];
22  k -= a[7];
23  k += a[6];
```

Source lines with Read and Write 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

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization: Loop Type	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
[loop in std::_Complex_base<double,struct _C_double_complex>::...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	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

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 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

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	New
P7	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	New

4. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cxx:1063	RAW:1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cxx:622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cxx:925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns | Correctness Report

ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cxx:637	lcals.exe	
P23	0; 0	Unit stride	runCRawLoops.cxx:638	lcals.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cxx:628	lcals.exe	

```
635    j2 = ( j2 & 64-1 ) ;
636    p[ip][0] += y[i2+32];
637    p[ip][1] += z[j2+32];
638    i2 += e[i2+32];
639    j2 += f[j2+32];
```

```
626    i1 &= 64-1;
627    j1 &= 64-1;
628    p[ip][2] += b[j1][i1];
```

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
                   //We need a way to check how we are
                   //accessing memory.
```

- Indirect reference in a loop

```
for (i=0;i<N;i++)
     A[B[i]] = C[i]*D[i];//We have to decode B[i] to find out
                 //which element of A to reference
```

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					Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op ...		Assign vector ...	0,013s	12,020s	Collapse	Collapse
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	4 Serialized use ...	0,013s	11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	2 Data type co ...	0,000s	0,163s	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o ...	☑	2 Data type co ...	0,000s	0,576s	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper ...	☐	2 Data type co ...	0,010s	12,030s	Scalar	

Select loops of interest

2.2 Check Memory Access Patterns

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

Command Line

Run Memory Access Patterns analysis, just to check how memory is used in the loop and the called function

Find vector optimization opportunities

Memory Access pattern analysis

Stride distribution

Check memory access patterns 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_79	operator()	fractal.cpp:179	No information available	100% / < 1,0000% / ...	Mixed strides
loop_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	100% / 0% / 0%	All unit strides

All memory accesses are uniform, with zero unit stride, so the same data is read in each iteration

We can therefore declare this function using the omp syntax: `pragma omp declare simd uniform(x0`

Memory Access Patterns Report

ID		Stride	Type		
P18		0	Unit stride		
P21		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      fy0 = fy0 / magn + cy;
70
```

P27		0	Unit stride	fractal.cpp:69	fractal.exe
P30		0	Unit stride	fractal.cpp:74	fractal.exe

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

Check memory access patterns in your application — Intel Advisor XE 201

Summary | Survey Report | Refinement Reports | MAP Source: fractal.cpp | Annotation Report | Suitability Report

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_54	operator()	fractal.cpp:164	No dependencies found	No information available	No information available
loop_site_129	operator()	fractal.cpp:164	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns Report | Correctness Report

ID	Stride	Type	Source	Modules	Alignment
P1		Parallel site information	fractal.cpp:164	fractal.exe	
P3	0	Unit stride	fractal.cpp:100	fractal.exe	
P4	0	Unit stride	fractal.cpp:164	fractal.exe	
P5	0	Unit stride	fractal.cpp:164	fractal.exe	
P6	0; 1	Unit stride	fractal.cpp:165	fractal.exe	
P7	0; 1	Unit stride	fractal.cpp:165	fractal.exe	
P8	0	Unit stride	fractal.cpp:60	fractal.exe	

```
98        }
99    #endif
100       int b = (int)(256 * mu);
101       int g = (b / 8);
102       int r = (g / 16);
```

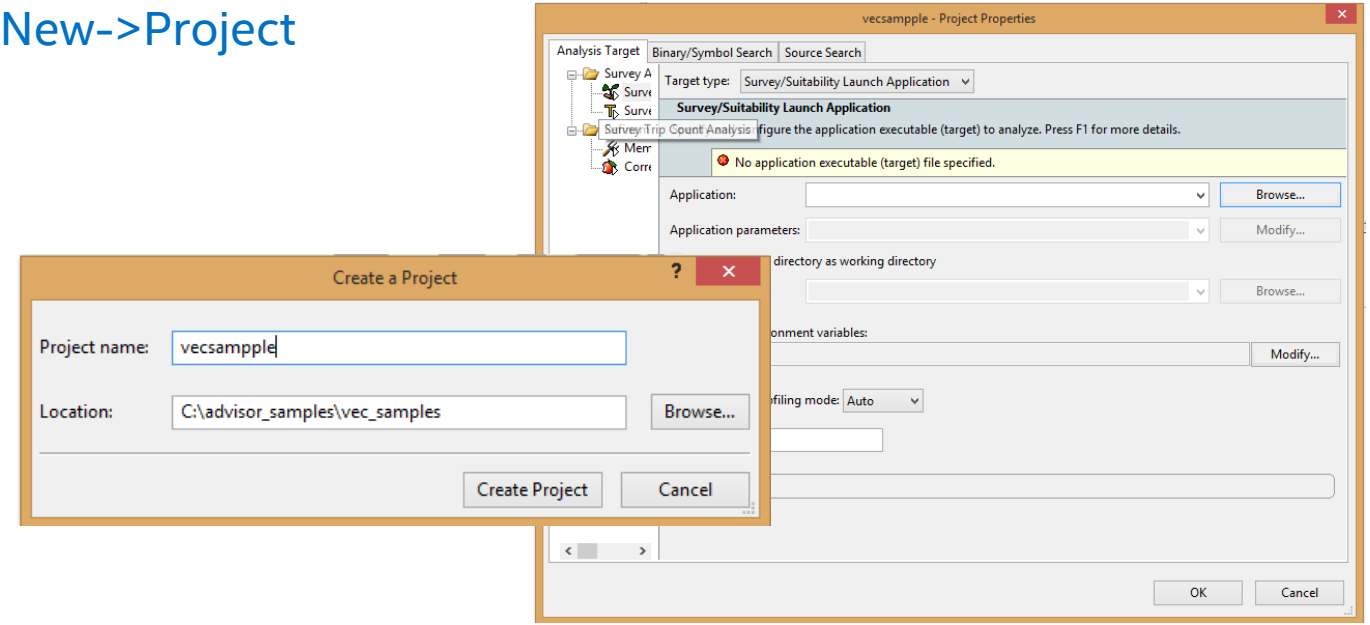
```
162
163                 for (int x = x0; x < x1; ++x) {
164                     for (int y = y0; y < y1; ++y) {
165                         fractal_data_array[x - x0][y - y0] = calc_one_pixel(x, y, tmp_max_iterations, tmp_size_x, tmp_si
166                     }
```

GETTING STARTED

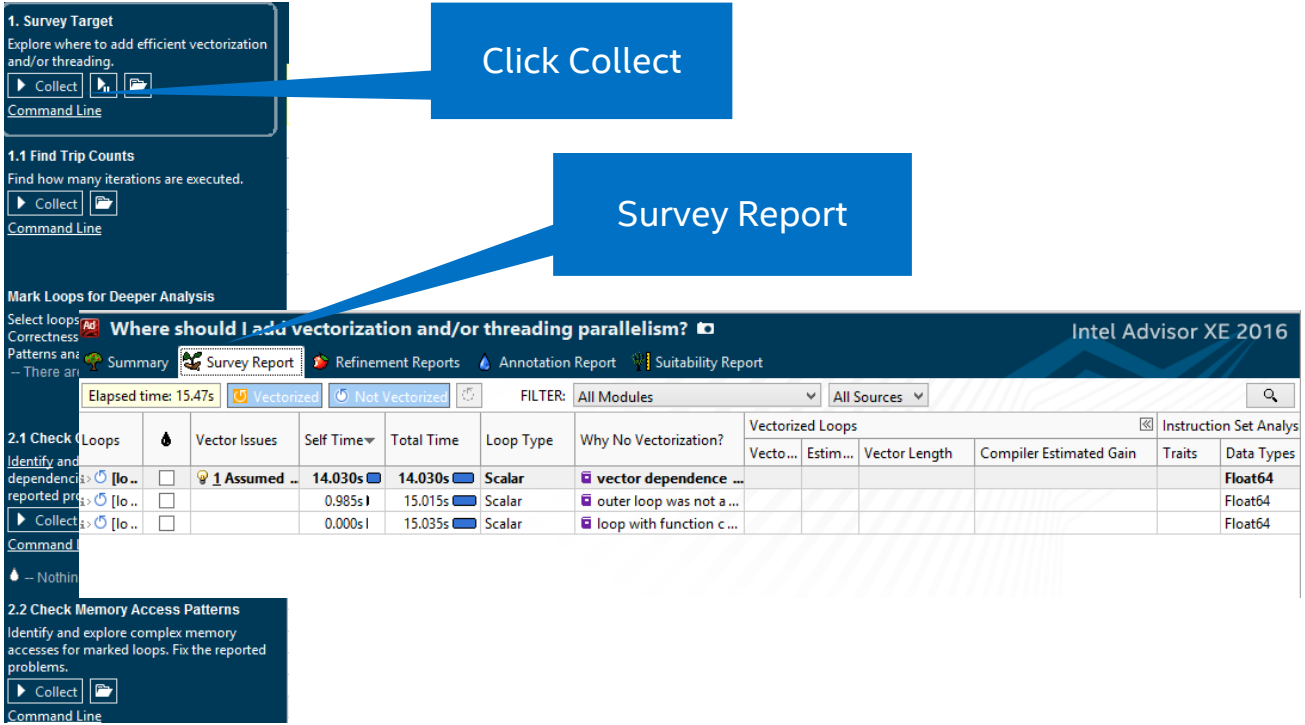
Before you analyze

Create Project

- File->New->Project



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



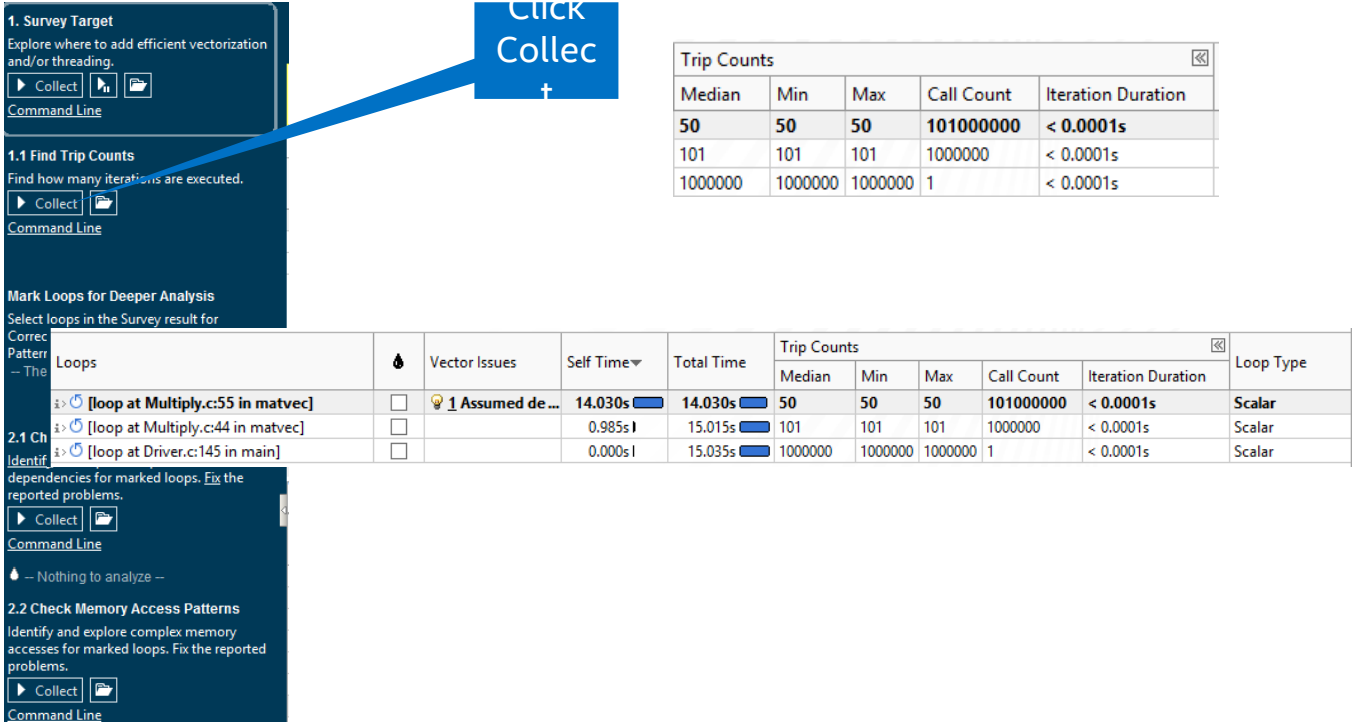
All the data in one place

Top Down | Source | Assembly | Recommendations | Compiler Diagnostics

Top Down | Source | Loop Assembly | Assistance | Recommendations | Compiler Diagnostic Details

Function Call Sites and Loops	Total Time %	Total Time	Self Time	Loop Type	Why No Vectorization?	Vectorized Loops			Instru
						Vecto...	Vector Length	Compiler Estimated Gain	Traits
Total	100.0%	15.043s	0s						
func@0x6b2daccf	100.0%	15.043s	0s						
func@0x6b2dacf0	100.0%	15.043s	0s						
BaseThreadInitThunk	100.0%	15.043s	0s						
_tmainCRTStartup	100.0%	15.043s	0s						
main	99.9%	15.035s	0s						
[loop at Driver.c:145 i	99.9%	15.035s	0s	Scalar	loop with function call ...				
printf	0.0%	0.001s	0.0006s						
func@0x10150ef0	0.1%	0.008s	0.0076s						

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



Specify loops for deeper analysis

Where should I add vectorization and/or threading parallelism?

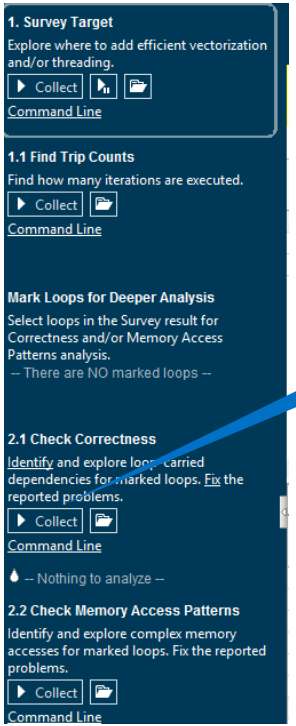
Summary | Survey Report | Refinement Reports | Annotation Report | Suitability Report

Elapsed time: 15.47s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Loops		Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at Multiply.c:55 in matvec]	✓	1 Assumed ...	14.030s	14.030s	Scalar	vector dependence p ...
[loop at Multiply.c:44 in matvec]	✓		0.985s	15.015s	Scalar	outer loop was not a ...
[loop at Driver.c:145 in main]	✓		0.000s	15.035s	Scalar	loop with function c ...

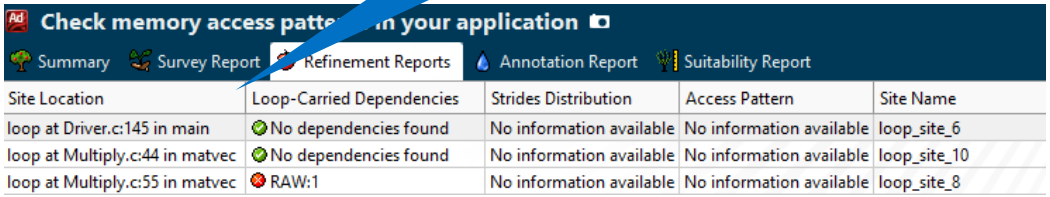
Deeper analysis

Check dependencies



Click Collect

We marked 3 loops for a dependency analysis. Two of the loops had no dependencies. One of the loops has Read-After-Write dependency and can't be vectorized.



Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Site Name
loop at Driver.c:145 in main	No dependencies found	No information available	No information available	loop_site_6
loop at Multiply.c:44 in matvec	No dependencies found	No information available	No information available	loop_site_10
loop at Multiply.c:55 in matvec	RAW:1	No information available	No information available	loop_site_8

Deeper analysis

Memory Access Pattern analysis

Stride distribution

1. 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

Command Line

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. Fix the reported problems.

Collect

Command Line

-- 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 Report | Refinement Reports | An ation Report | Suitability Report

Site Location	Loop-Carried Dependencies	Strides 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	85% / 15% / 0%	Mixed strides	loop_site_10
loop at Multiply.c:55 in matvec	RAW:1	74% / 26% / 0%	Mixed strides	loop_site_8

Memory Access Patterns Report | Correctness Report

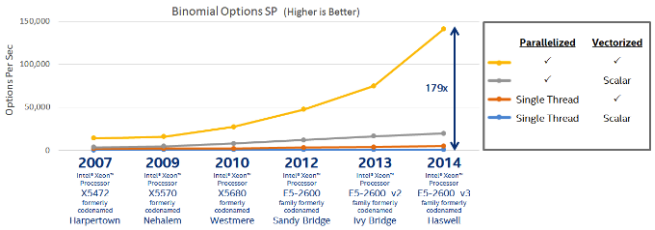
ID		Stride	Type	Source	Nested Function	Modules	Alignment
P3			Parallel site information	Driver.c:145		matrix_vector_multiplication_c.exe	
P9		0	Unit stride	Driver.c:157		matrix_vector_multiplication_c.exe	
P10		0	Unit stride	Multiply.c:39	matvec	matrix_vector_multiplication_c.exe	
P12		0	Unit stride	Multiply.c:44	matvec	matrix_vector_multiplication_c.exe	
P14		0; 1	Unit stride	Multiply.c:45	matvec	matrix_vector_multiplication_c.exe	

```
43      int i, j;
44
45      for (i = 0; i < size1; i++) {
46          b[i] = 0;
47
```

Click Collect

intel®

Configurations for Binomial Options SP



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

Platform Hardware and Software Configuration

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

Legal Disclaimer & Optimization Notice

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.

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

intel®

experience
what's inside™