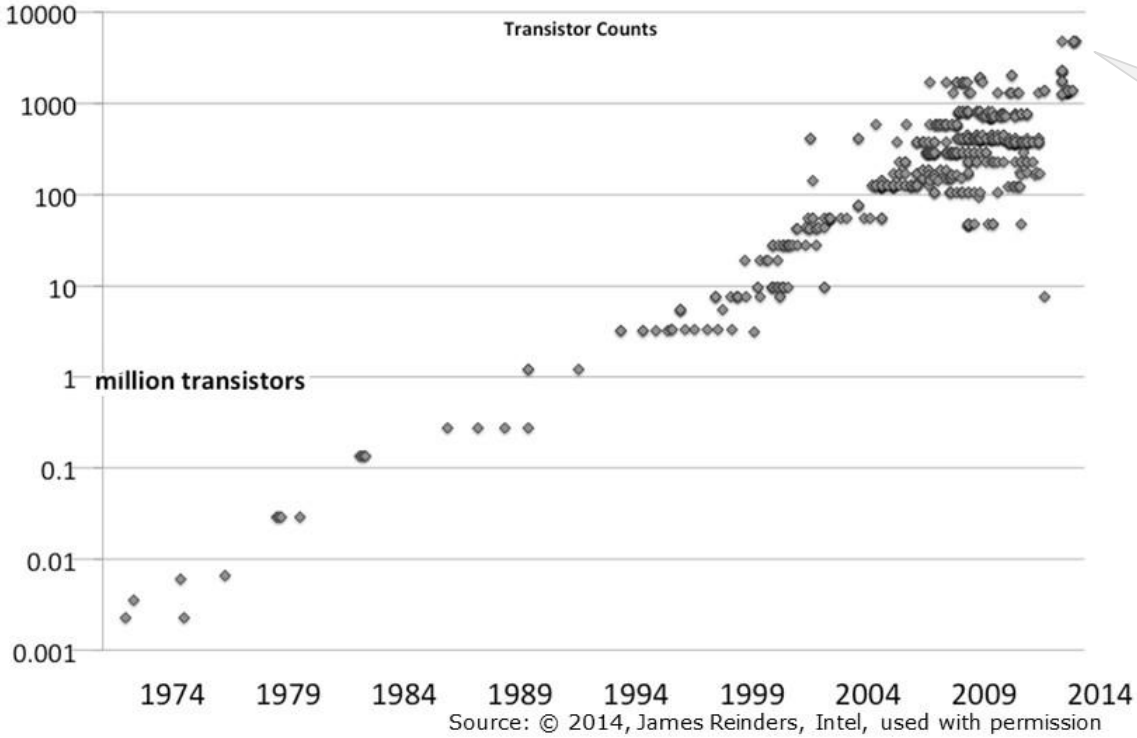
VECTORIZE OR PERFORMANCE DIES: TUNE FOR THE LATEST AVX SIMD

Kevin O'Leary, Intel Technical Consulting Engineer

Moore's Law Is Going Strong

Hardware performance continues to grow exponentially



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

"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	

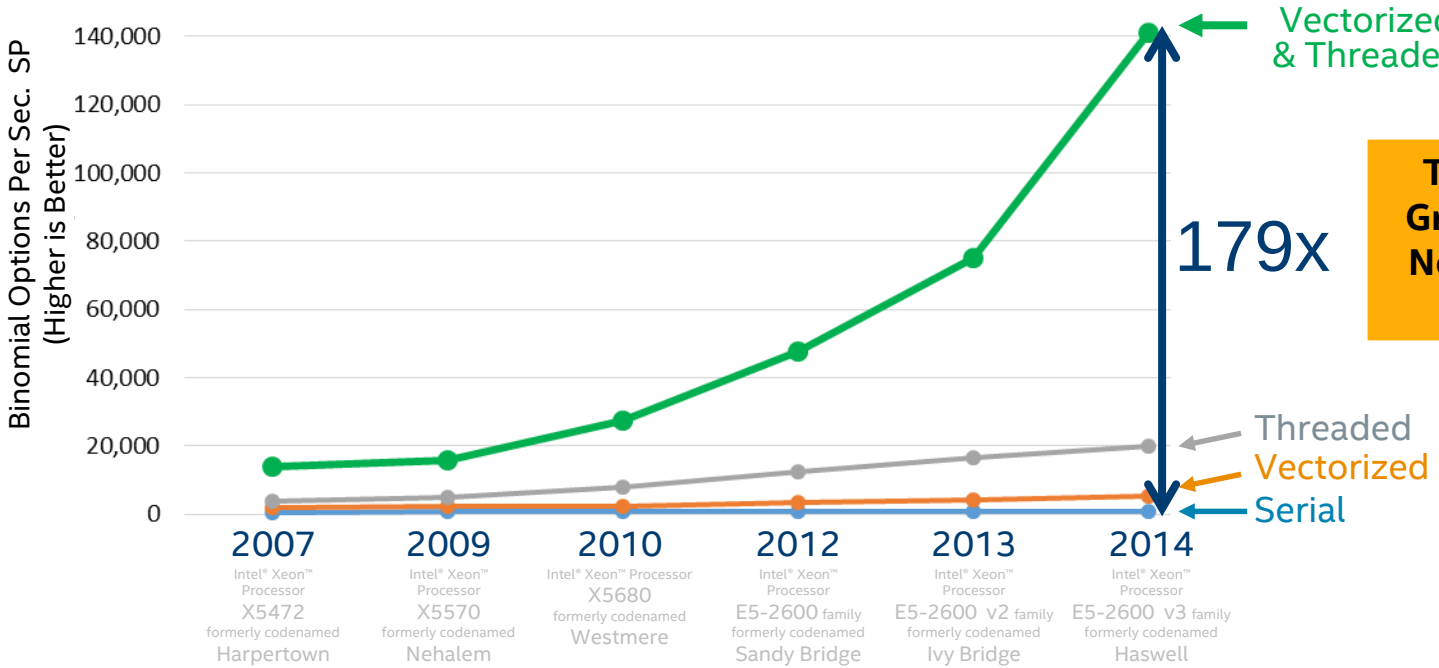
*Product specification for launched and shipped products available on ark.intel.com. 1. Not launched or in planning.

High performance software must be both:

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

Vectorize & Thread or Performance Dies

Threaded + Vectorized can be much faster than either one alone



The Difference Is Growing With Each New Generation of Hardware

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

INTEL® ADVISOR
VECTORIZATION OPTIMIZATION AND THREAD PROTOTYPING

Software Must Vectorize & Thread or Performance Dies

True today – More true tomorrow – Difference can be substantial!

Vectorization - Have you:

- Recompiled for AVX2 or AVX512 with little gain
- Wondered where to vectorize?
- Recoded intrinsics for new arch.?
- Struggled with compiler reports? New!

Threading - Have you:

- Threaded an app, but seen little benefit?
- Hit a "scalability barrier"?
- Delayed release due to sync. errors?

"**Intel® Advisor's** Vectorization Advisor fills a gap in code performance analysis. It can guide the informed user to better exploit the vector capabilities of modern processors and coprocessors."

Dr. Luigi Iapichino
Scientific Computing Expert
Leibniz Supercomputing Centre

"**Intel® Advisor** has allowed us to quickly prototype ideas for parallelism, saving developer time and effort"

Simon Hammond
Senior Technical Staff
Sandia National Laboratories

Here is What Will Be Covered

Overview

5 steps to Efficient Vectorization

Vectorization Efficiency

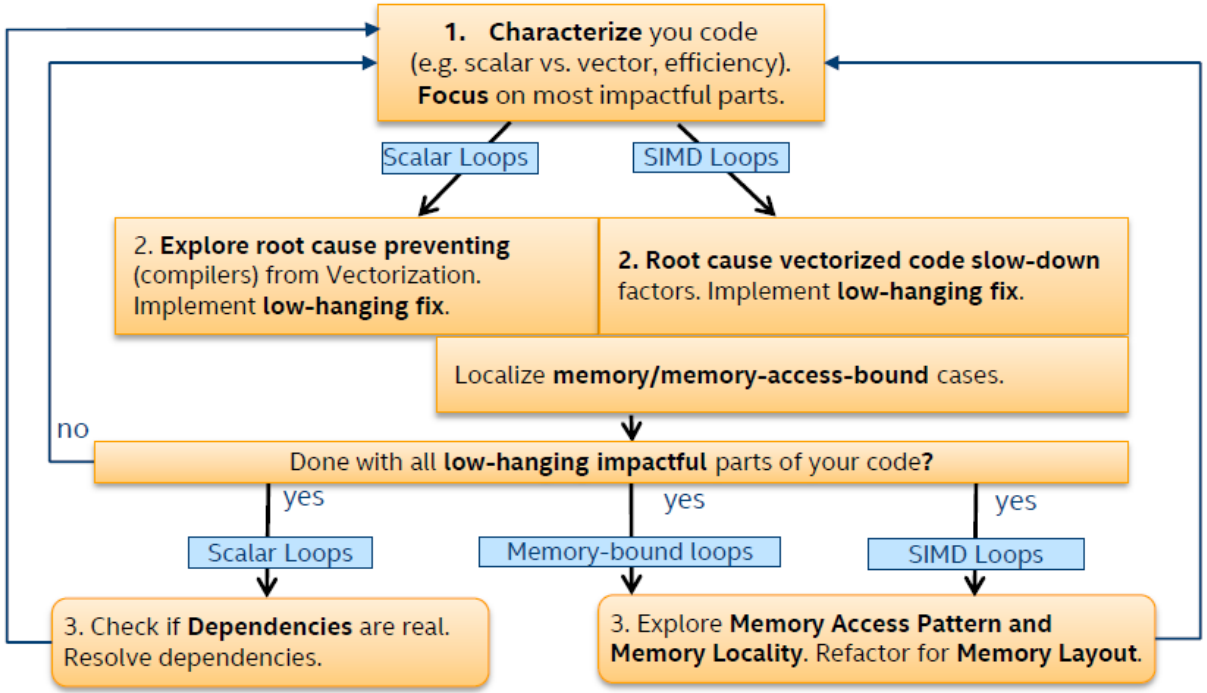
Background on Vectorization

New Features

Call to Action

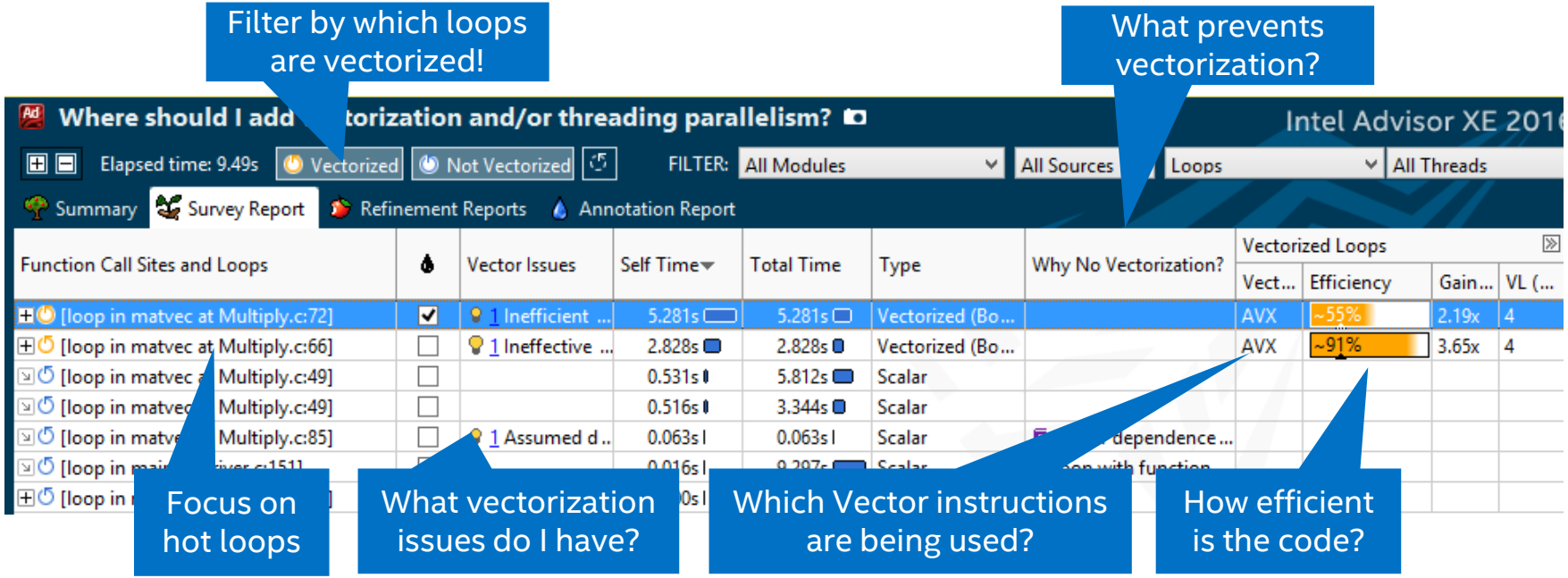
Intel® Advisor helps you increase performance!

Recommended methodology



The Right Data At Your Fingertips

Get all the data you need for high impact vectorization



Get Fast Code Fast!

5 Steps to Efficient Vectorization - Vector Advisor

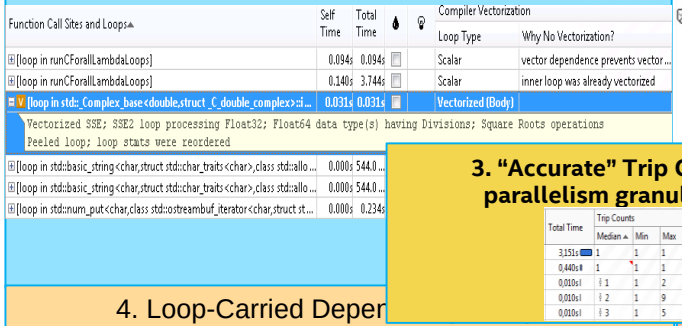
1. Compiler diagnostics + Performance Data + SIMD efficiency information

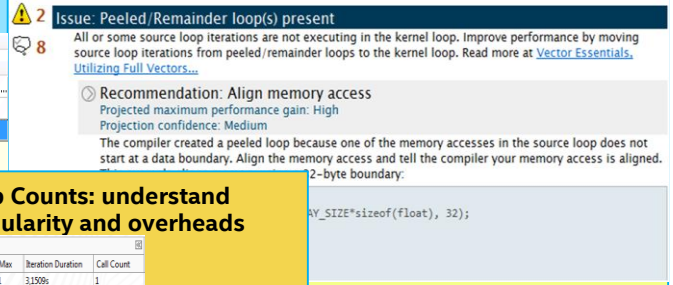
2. Guidance: detect problem and recommend how to fix it

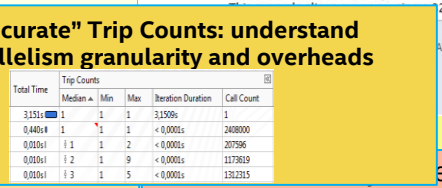
3. "Accurate" Trip Counts: understand parallelism granularity and overheads

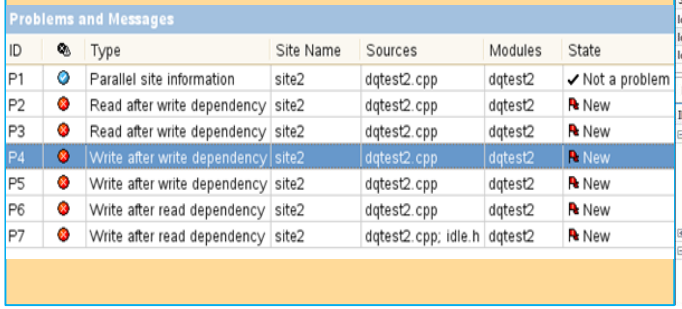
4. Loop-Carried Deper

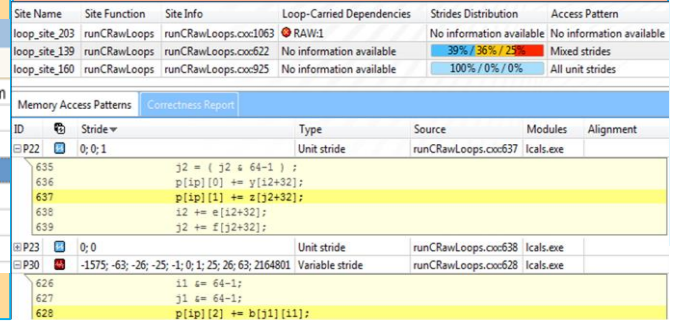
ess Patterns Analysis





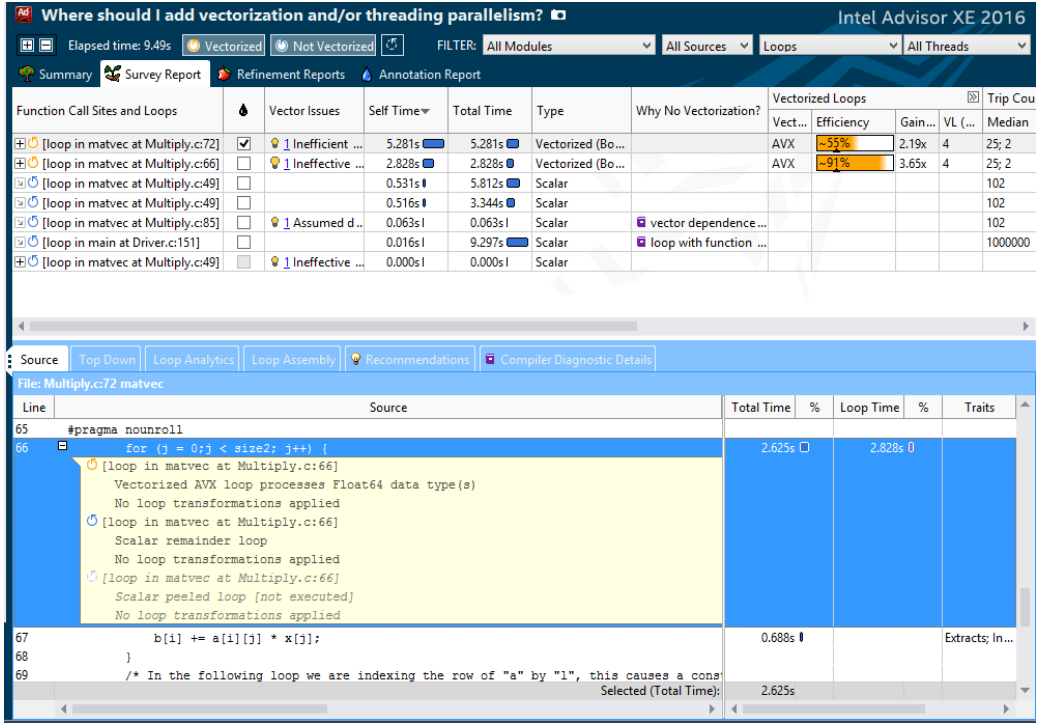




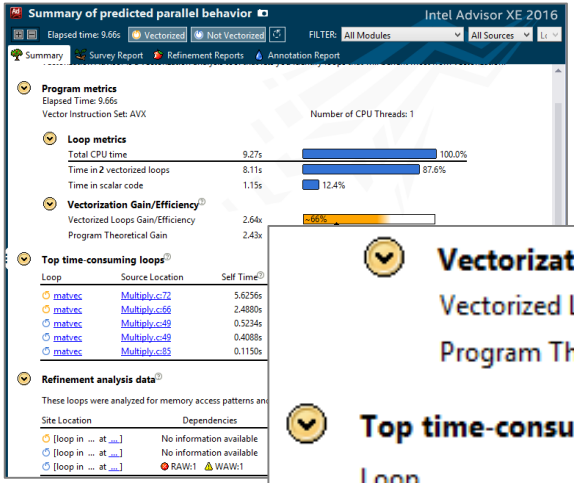


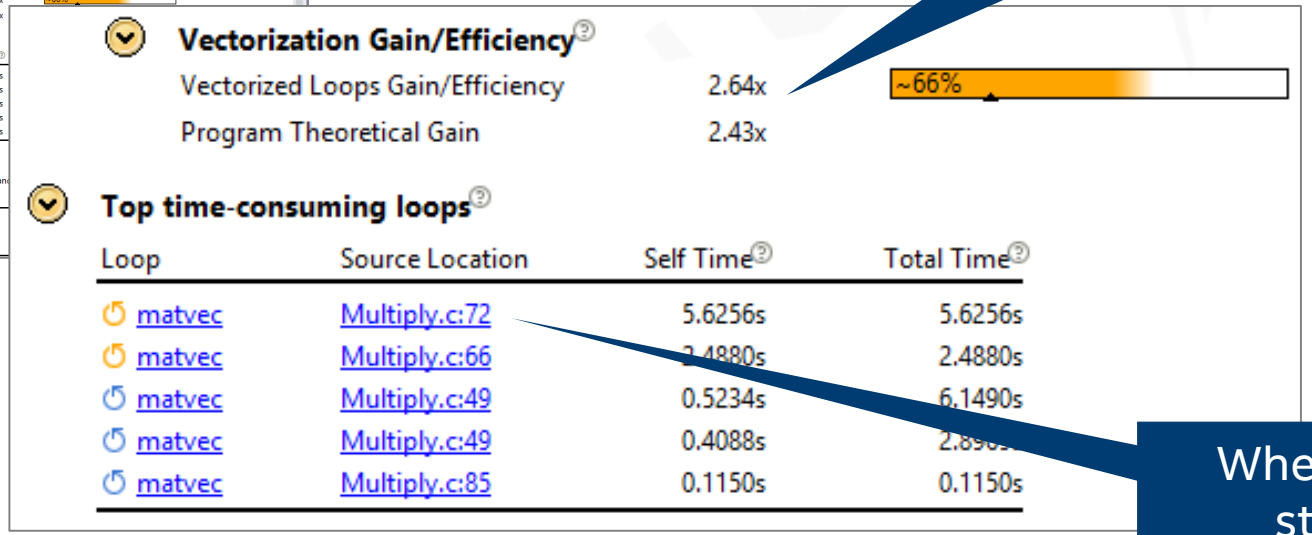
Part of Intel® Advisor, Intel® Parallel Studio XE 2016

Efficiently Vectorize your Code
Intel Advisor – Vectorization Advisor



Summary View: Plan Your Next Steps



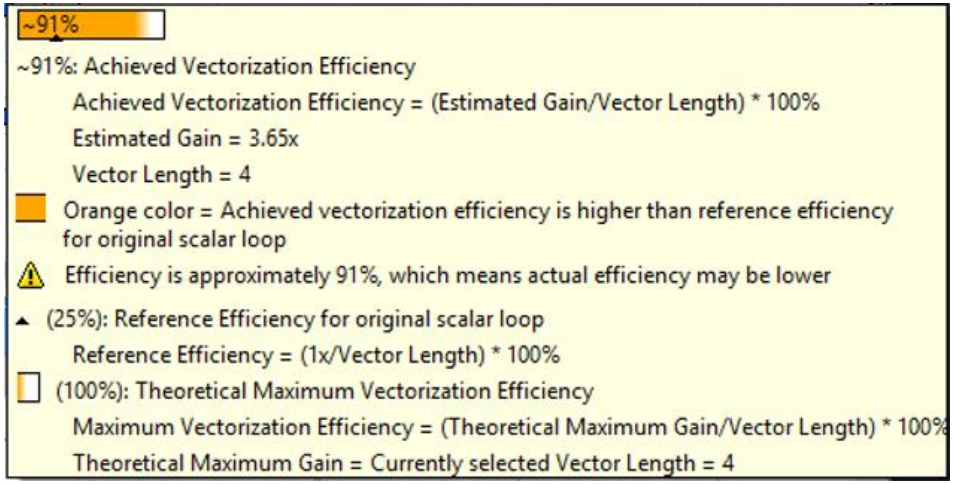


What can I expect to gain?

Where do I start?

Vector Efficiency: All The Data In One Place
My "performance thermometer"

Vector Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops				Trip Counts	Instruction Set Analysis		
					Vect...	Efficiency	Gain...	VL (...		Traits	Data T...	Num...
1 Inefficient ...	5.281s	5.281s	Vectorized (Bo...		AVX	~55%	2.19x	4	25; 2	Inserts	Float64	3; 7
1 Ineffective ...	2.828s	2.828s	Vectorized (Bo...		AVX	~91%	3.65x	4	25; 2		Float64	3



Look at Vector Issues and Traits to find out why

- All kinds of "memory manipulations"
- Usually an indication of "bad" access pattern

Spend your time in the most efficient place! 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! (and you tell the compiler it is aligned!)
- Make the number of iterations divisible by the vector length!

6 Factors That Impact Vectorization Efficiency

Loop-carried dependencies

```
DO I = 1, N
   A(I+1) = A(I) + B(I)
ENDDO
```

Function calls

```
for (i = 1; i < nx; i++) {
  x = x0 + i * h;
  sumx = sumx + func(x, y, xp);
}
```

Pointer aliasing

```
void scale(int *a, int *b)
{
   for (int i = 0; i < 1000; i++)
          b[i] = z * a[i];
}
```

Unknown loop iteration count

```
struct _x { int d; int bound; };

void doit(int *a, struct _x *x)
{
  for(int i = 0; i < x->bound; i++)
    a[i] = 0;
}
```

Indirect memory access

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

Outer loops

```
for(i = 0; i <= MAX; i++) {
  for(j = 0; j <= MAX; j++) {
    D[i][j] += 1;
  }
}
```

many .....

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.

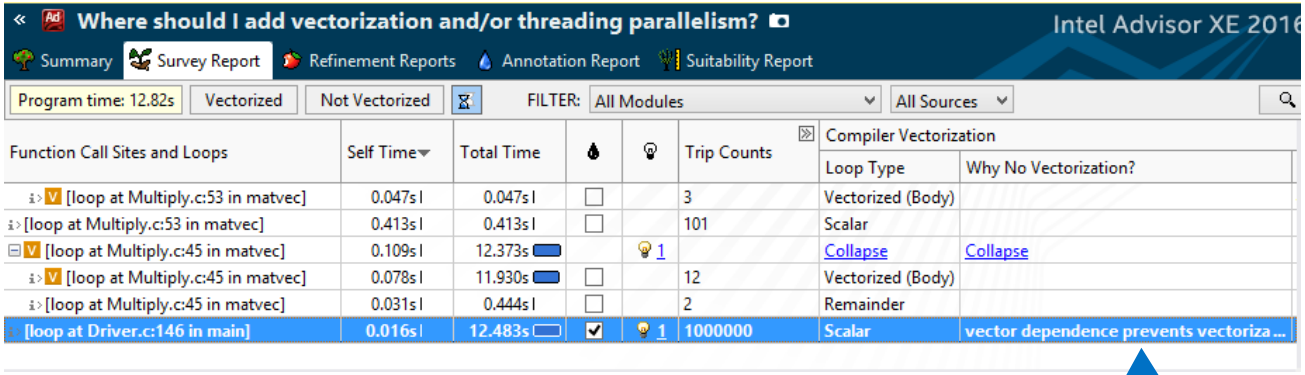
ICL/ICC/ICPC Directive	IFORT Directive	Outcome
#pragma simd or #pragma omp simd	!DIR$ SIMD or !$OMP SIMD	Ignores all dependencies in the loop
#pragma ivdep	!DIR$ IVDEP	Ignores only vector dependencies (which is safest)

Read More:

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

Check if It Is It Safe to Vectorize

Loop-Carried Dependencies Analysis Verifies Correctness



2.1 Check Dependencies

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

Collect

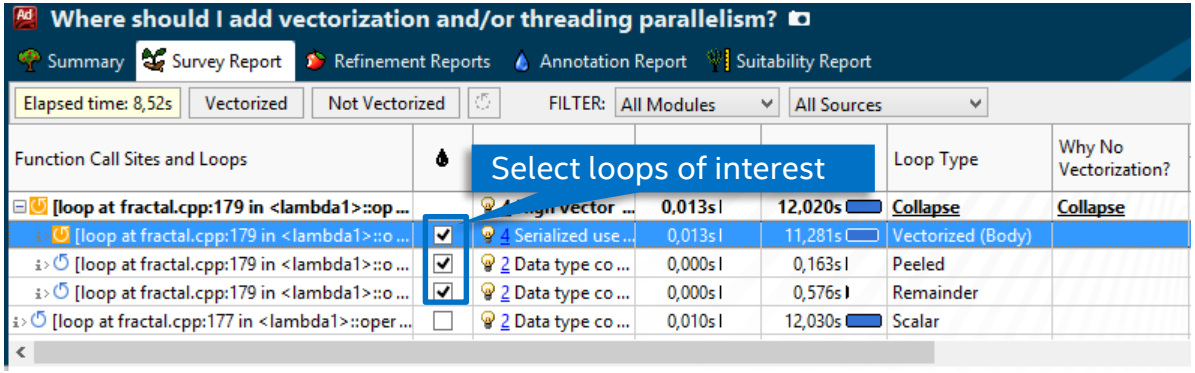
Command Line

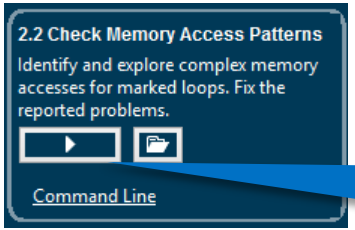
Select loop for Dependency Analysis and press play!

Vector Dependence prevents Vectorization!

Improve Vectorization

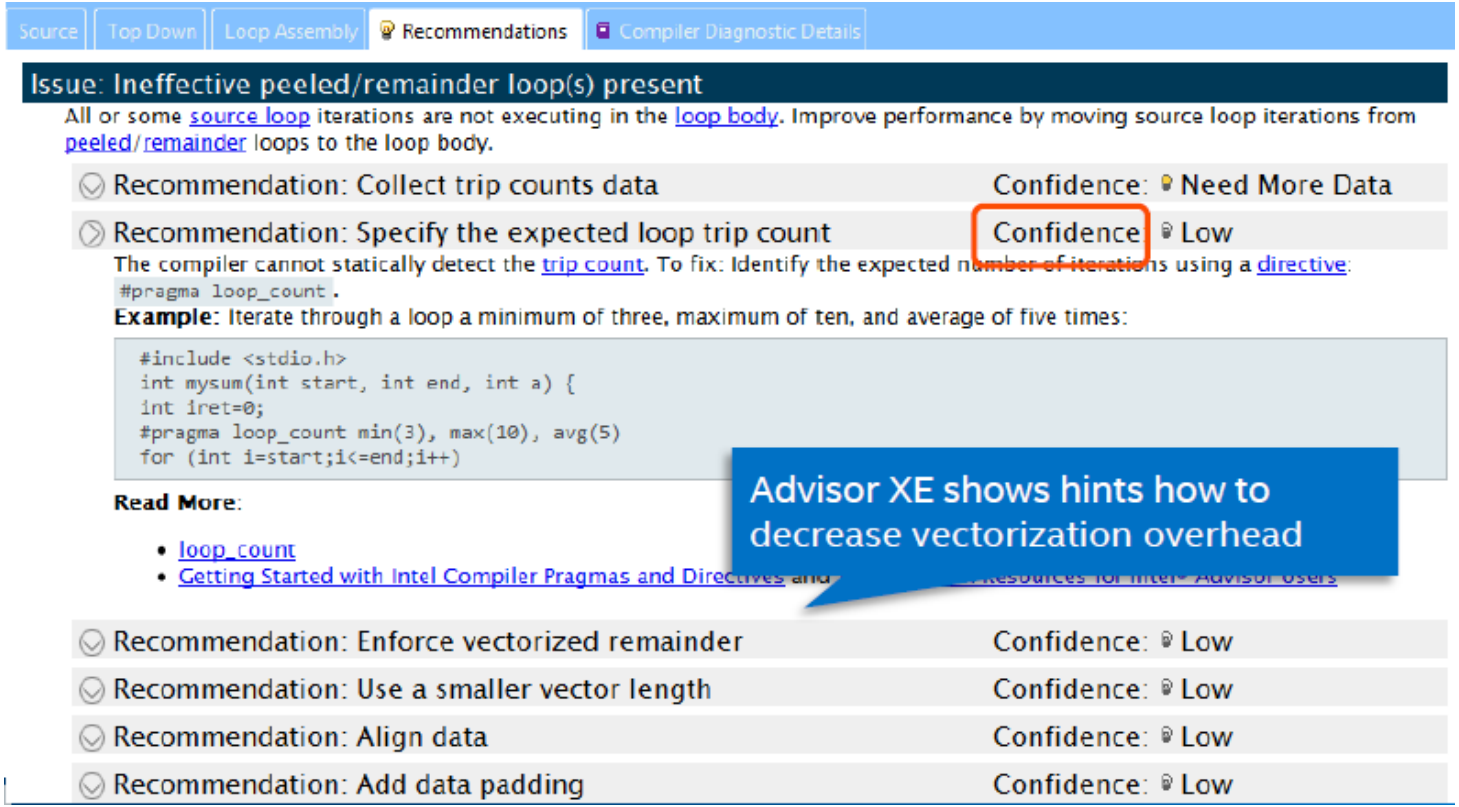
Memory Access Pattern Analysis





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

Get specific advice for Improving Vectorization



New Features in Intel® Advisor

Next generation Intel® Xeon Phi™ processor (code named Knights Landing)

Explore non-executed code paths

- Generate multiple code paths for instructions that your machine does not even support (for example AVX-512)

Batch mode workflow

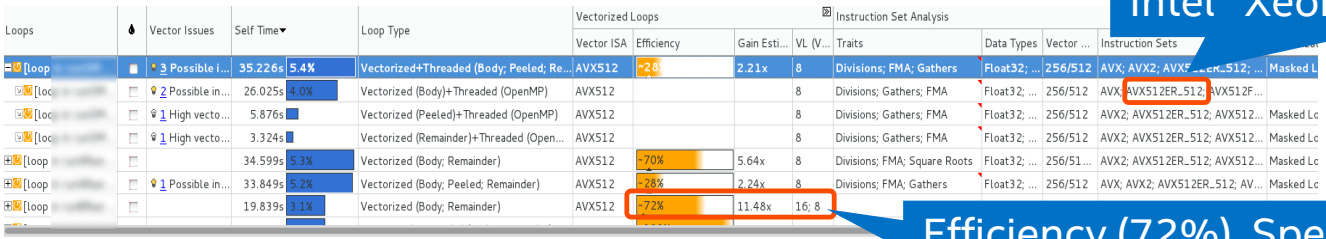
Filter by thread

Loop Analytics

Gather instruction profiling

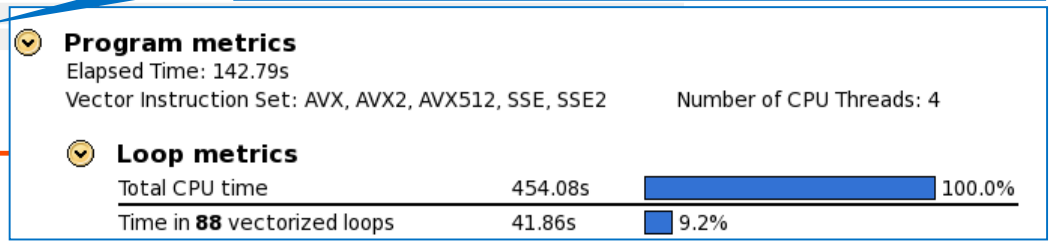
Vectorization Advisor runs on and optimizes for Intel® Xeon Phi™ architecture

AVX-512 ERI – specific to Intel® Xeon Phi



Efficiency (72%), Speed-up (11.5x), Vector Length (16)

Performance optimization problem and advice how to fix it



Start Tuning for AVX-512 without AVX-512 hardware

Intel® Advisor - Vectorization Advisor

Use –axCOMMON-AVX512 –xAVX compiler flags to generate both code-paths

- AVX(2) code path (executed on Haswell and earlier processors)
- AVX-512 code path for newer hardware

Compare AVX and AVX-512 code with Intel Advisor

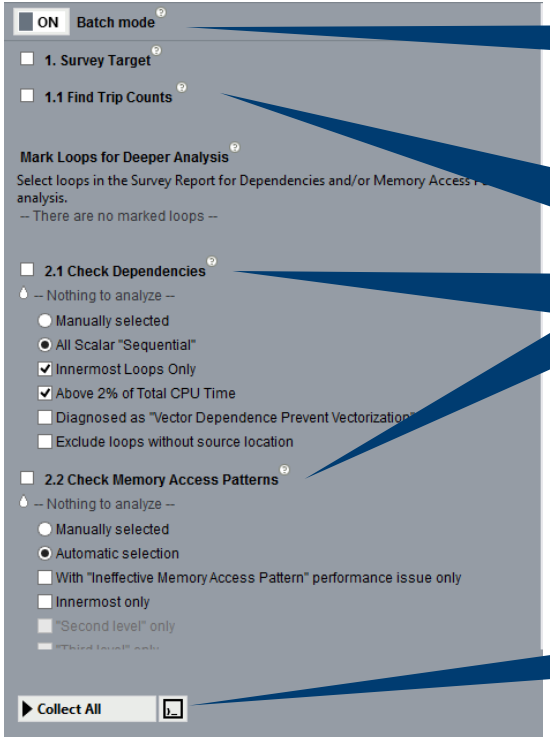
Loops		Self Time	Loop Type	Vectorized Loops					Instruction Set Analysis				Advanced
				Vect...	Efficiency	Gain...	VL (...	Compiler Es...	Traits	Data T...	Vector W...	Instruction Sets	Vectorization De
[loop in s352_ at loopstl.cpp:5939]		**0,641s**	**Vectorized (Body)**	**AVX2**	**~54%**	**2,15x**	**4**	**2,15x**	**FMA; Inserts**	**Float32**	**128**	**AVX; FMA**	
[loop in s352_ at loopstl.cpp:5939]		n/a	Remainder [Not Executed]				4		FMA				
[loop in s352_ at loopstl.cpp:5939]		0,641s	Vectorized (Body)	AVX2			4	2,15x	Inserts; FMA				
[loop in s352_ at loopstl.cpp:5939]		n/a	Vectorized (Body) [Not Executed]	AVX512			16	3,20x	Gathers; FMA				
[loop in s352_ at loopstl.cpp:5939]		n/a	Vectorized (Remainder) [Not Executed]	AVX512			16	2,70x	Gathers; FMA				
[loop in s125_.Aompparallel_for@ ...		0,496s	Vectorized Versions	AVX2	~100%	13,54x	8	<13,54x	FMA; NT-stores				
[loop in s125_.Aompparallel_for...		n/a	Peeled [Not Executed]				8		FMA				
[loop in s125_.Aompparallel_for...		n/a	Remainder [Not Executed]				8		FMA				
[loop in s125_.Aompparallel_for...		0,465s	Vectorized (Body)	AVX2			8	13,54x					
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Peeled) [Not Executed]	AVX512			16	6,77x	FMA				
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Body) [Not Executed]	AVX512			32	30,61x					
[loop in s125_.Zompparallel_for...		n/a	Vectorized (Remainder) [Not Executed]	AVX512			16	9,78x	FMA				

Inserts (AVX2) vs. Gathers (AVX-512)

Speed-up estimate: 13.5x (AVX2) vs. 30.6x (AVX-512)

Batch Mode Workflow Saves Time

Intel® Advisor - Vectorization Advisor



Turn On Batch Mode

Select analyses to run

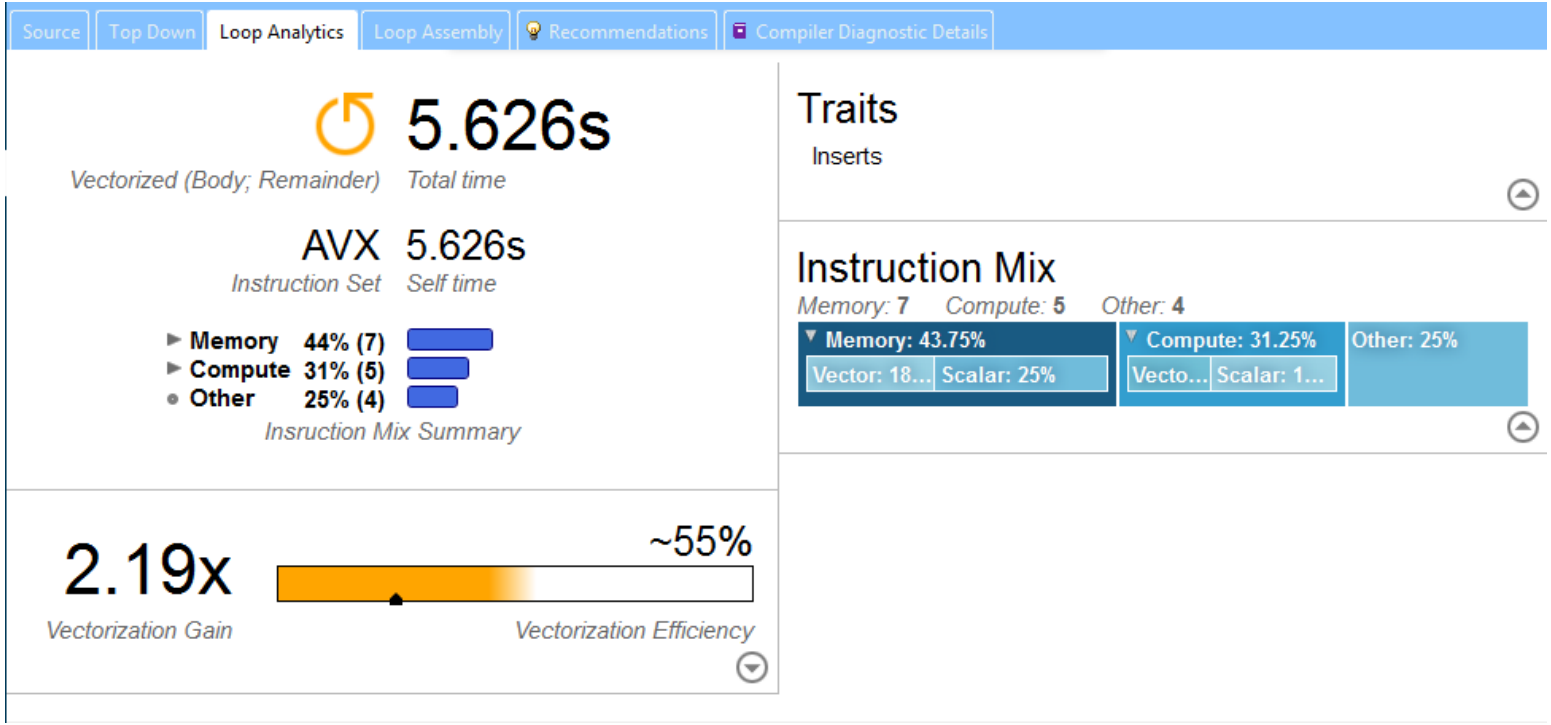
Click Collect all

Run several analyses in batch as a single run

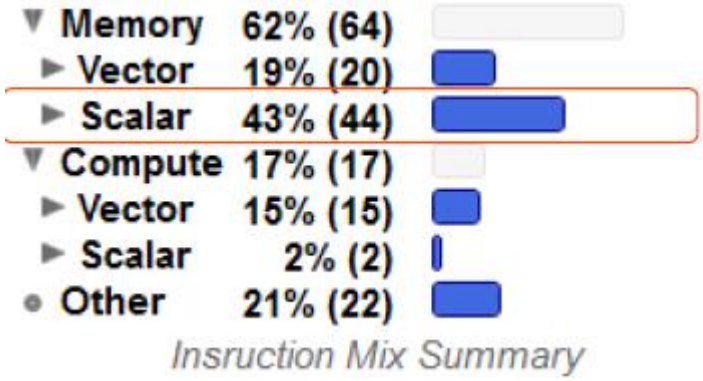
Contains pre-selected criteria for advanced analyses

Loop Analytics

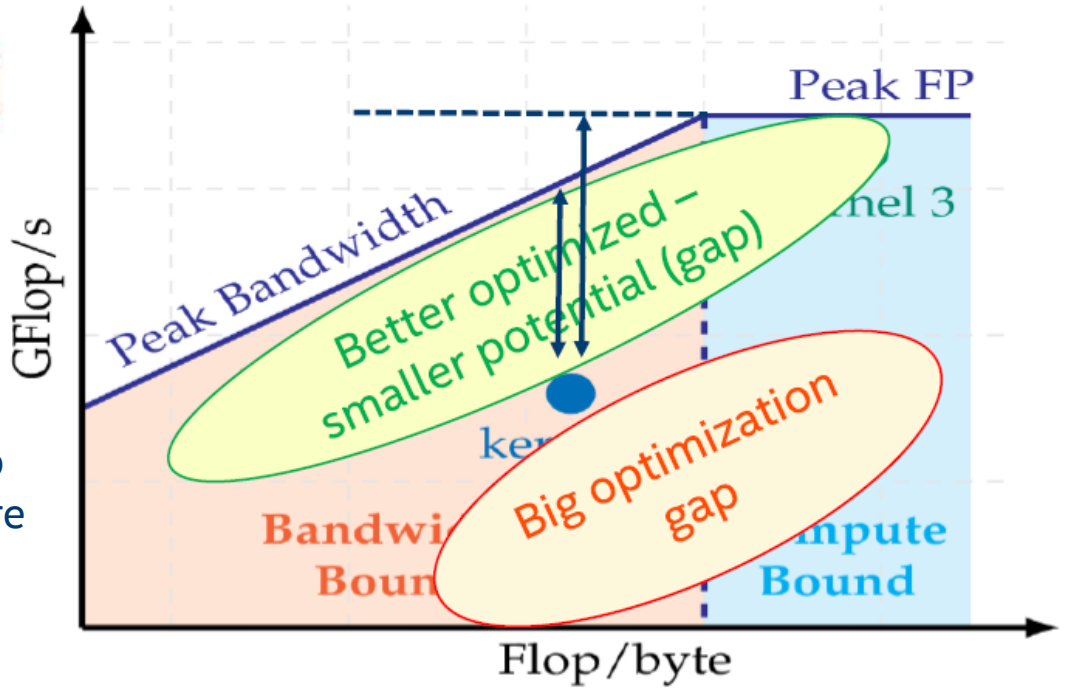
Get detailed information about your loops



Am I memory bound or VPU/CPU bound?



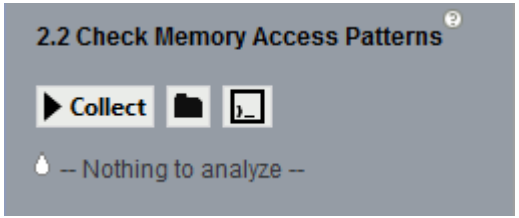
The types of instructions in your loop will be an indicator of whether your are memory bound or compute bound.

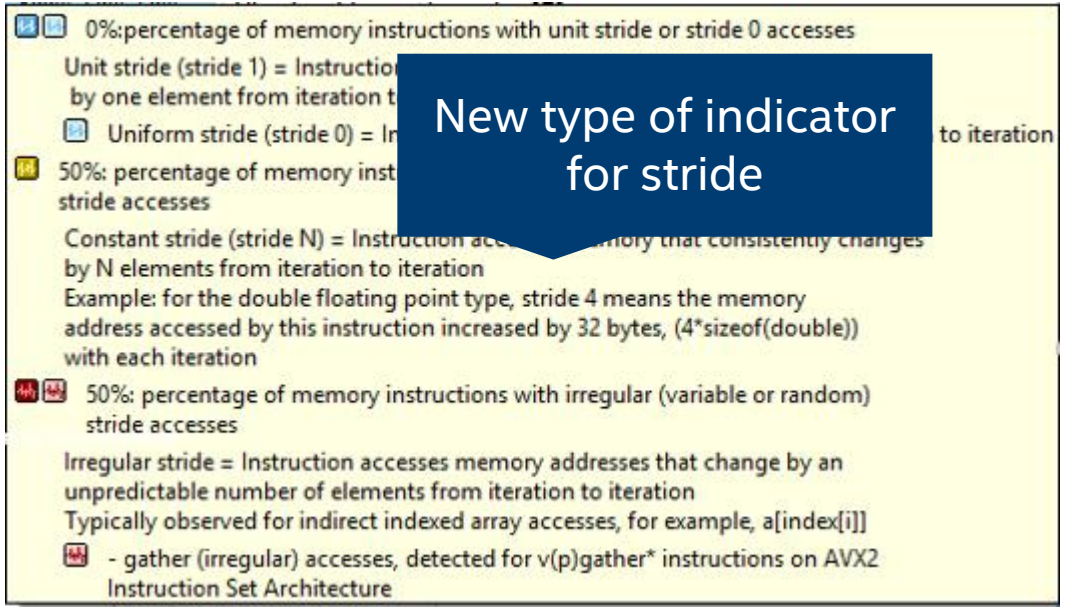


Irregular access patterns decreases performance!

Gather profiling

Run Memory Access Pattern Analysis





Gather/Scatter analysis is very important for AVX512

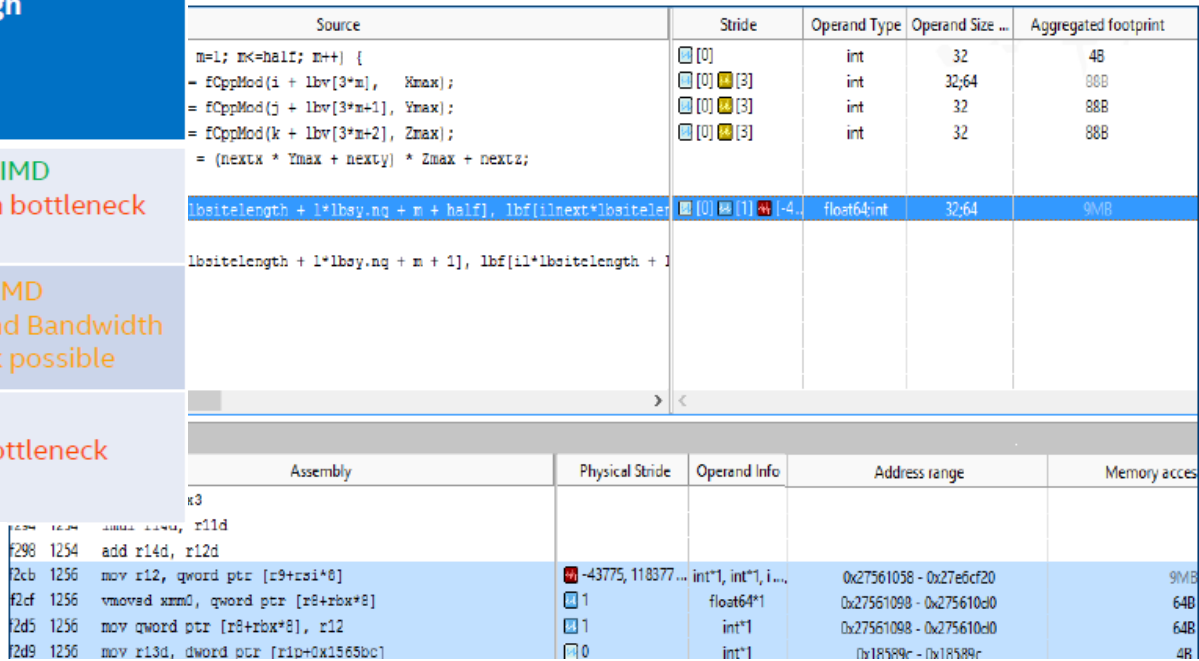
AVX512 Gather/Scatter in wider use than on previous instruction sets

- Many more applications can now be vectorized
- Gives good average performance but far from optimal
- Much greater need for Gather/Scatter profiling
- With Intel® Advisor you get both dynamic and static gather/scatter information

Memory Analysis Is Critical

Determine Possible Bandwidth or Latency Issues

Footprint / Access Pattern	Small enough	Big enough
Unit Stride	Effective SIMD No Latency and BW bottlenecks	Effective SIMD Bandwidth bottleneck
Const stride	Medium SIMD Latency bottleneck possible	Medium SIMD Latency and Bandwidth bottleneck possible
Irregular Access, Gather/Scatter	Bad SIMD Latency bottleneck possible	Bad SIMD Latency bottleneck



Detect possible latency or bandwidth issues

Enhanced Memory Analysis

Address	Line	Physical Stride	Operand Info	Vector Length	Operand Size (bits)	Address range	Memory access fo...
0x14002eabd							
0x14002eac1							
0x14002eac7							
0x14002eacb							
0x14002ead0							
0x14002ead0		2	float32*8	8	256	0x23a2d960 - 0x23a ...	288B
0x14002ead6		2	float32*8	8	256	0x23a2d980 - 0x23a ...	288B
0x14002eadd							
0x14002eae1							
0x14002eae5			float32*8;int ...	8	32	0x87af10 - 0x87b038	300B
0x14002eaec							
0x14002eaf0							
0x14002eaf4			float32*8;int ...	8	32	0x87af28 - 0x87	
0x14002eafb		2	float32*8	8	256	0x879f70 - 0x87	
0x14002eb01		2	float32*8	8	256	0x879f90 - 0x87	

Is your memory range small enough to fit in cache?

Gather (irregular) access

Operand Size (bits): 32

Operand Type: float32,int32

Vector Length: 8

Memory access footprint: 300B

Gather details

Mask is constant

Mask: [11111111]

Active elements in the mask: 100.0%

Mask Utilization and FLOPS profiler

- Long-waiting in HPC: accurate HW independent FLOPs measurement tool
- Not just count FLOPs. Has following additions:
  - (AVX-512 only) Mask-aware. Masked-Memory/Unmasked-Compute pattern aware
  - Unique capability to correlate FLOPs with performance data (obtained without instrumentation). Gives FLOPs/s.
- Lightweight instrumentation, PIN-based, benefits from “threadchecker tools” and more generally Advisor framework integration.

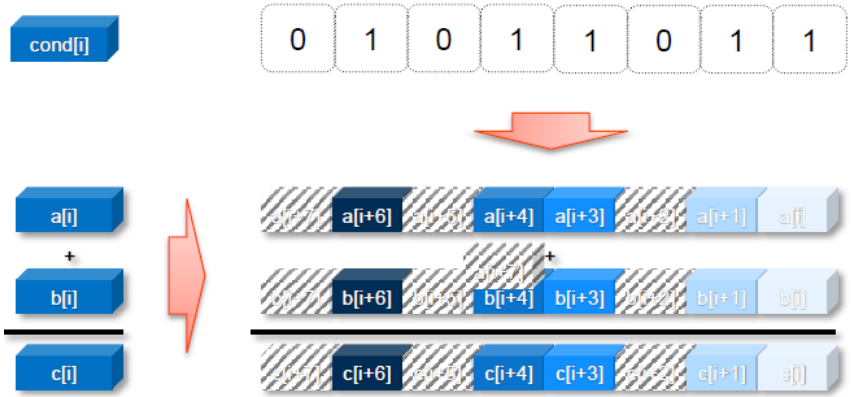
Why is Mask Utilization important?

3 elements suppressed

SIMD Utilization = 5/8 (62.5%)

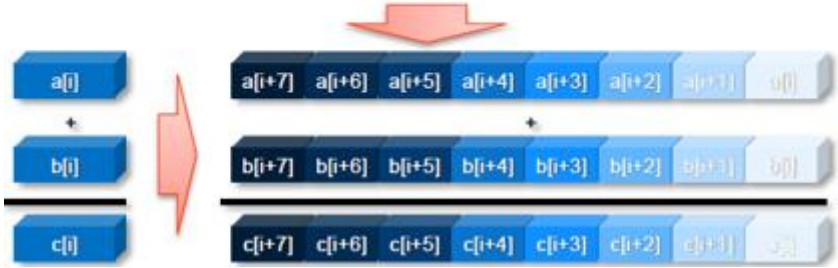
Not utilizing full vectors!!

```
for(i = 0; i <= MAX; i++)
   if (cond(i))
      c[i] = a[i] + b[i];
```



Fully utilized!

```
for(i = 0; i <= MAX; i++)
   c[i] = a[i] + b[i];
```

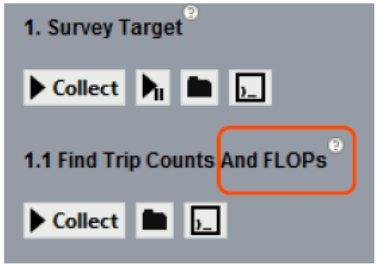


FLOPs and Mask Utilization Profiler

Tech Preview

set ADVIXE_EXPERIMENTAL=FLOPS

advixe-cl.exe --collect survey
advixe-cl.exe --collect trip counts



FLOPs, Masks, Trip Counts

Median	GFLOPs/s	Arithmetic Intensity	Mask Utiliz...	GBytes/s	GFLOP
19	2,456	0.125		19.6498	3.94488
4; 3	2,351	0.125	63,29%	18.8111	0.36693
19	2,136	0.0795455		26.8513	2.50206
19	1,910	0.0681818		28.011	1.07231
3	1,774	0.0833333		21.2898	0.11287
4	1,192	0.0666667		17.8726	0.1505
19	0,911	0.0681818		13.3635	0.0285

Call to Action

Modernize your Code

- To get the most out of your hardware, you need to modernize your code with vectorization and threading.
- Taking a methodical approach such as the one outlined in this presentation, and taking advantage of the powerful tools in Intel® Parallel Studio XE, can make the modernization task dramatically easier.
- Send e-mail to vector_advisor@intel.com to get the latest information on some exciting new capabilities that are currently under development.

Resources

Intel® Advisor Links

- Vectorization Guide
  - http://bit.ly/autovectorize-guide
- Explicit Vector Programming in Fortran
  - http://bit.ly/explicitvector-fortran
- Optimization Reports
  - http://bit.ly/optimizereports
- Beta Registration & Download
  - http://bit.ly/PSXE2017-Beta

Code Modernization Links

- Modern Code Developer Community
  - software.intel.com/modern-code
- Intel Code Modernization Enablement Program
  - software.intel.com/code-modernization-enablement
- Intel Parallel Computing Centers
  - software.intel.com/ipcc
- Technical Webinar Series Registration
  - http://bit.ly/spring16-tech-webinars
- Intel Parallel Universe Magazine
  - software.intel.com/intel-parallel-universe-magazine

Additional Resources

For Intel® Xeon Phi™ coprocessors, but also applicable:

- https://software.intel.com/en-us/articles/vectorization-essential
- https://software.intel.com/en-us/articles/fortran-array-data-and-arguments-and-vectorization

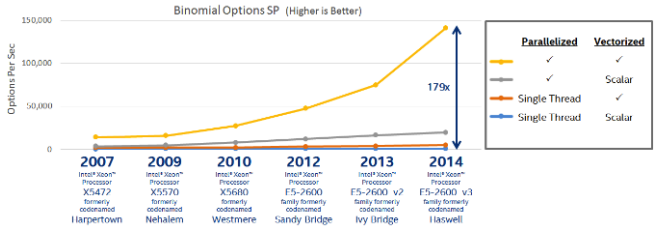
Intel® Parallel Studio XE Composer Edition User and Reference Guides:

- https://software.intel.com/en-us/intel-cplusplus-compiler-16.0-user-and-reference-guide-pdf
- https://software.intel.com/en-us/intel-fortran-compiler-16.0-user-and-reference-guide-pdf

Compiler User Forums

- http://software.intel.com/forums

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

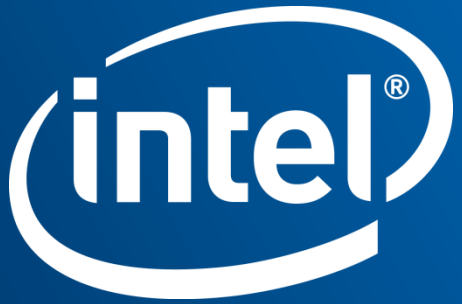
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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.

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®