USING INTEL® ADVISOR WITH THE ROOFLINE MODEL TO BOOST YOUR APPLICATIONS

Profiling and improving your vectorization

ACKNOWLEDGMENTS/REFERENCES

Roofline model proposed by Williams, Waterman, Patterson:
http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf

"Cache-aware Roofline model: Upgrading the loft" (Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon) http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf

At Intel:
Roman Belenov, **Zakhar Matveev**, Julia Fedorova
SSG product teams, Hugh Caffey,
in collaboration with **Philippe Thierry**

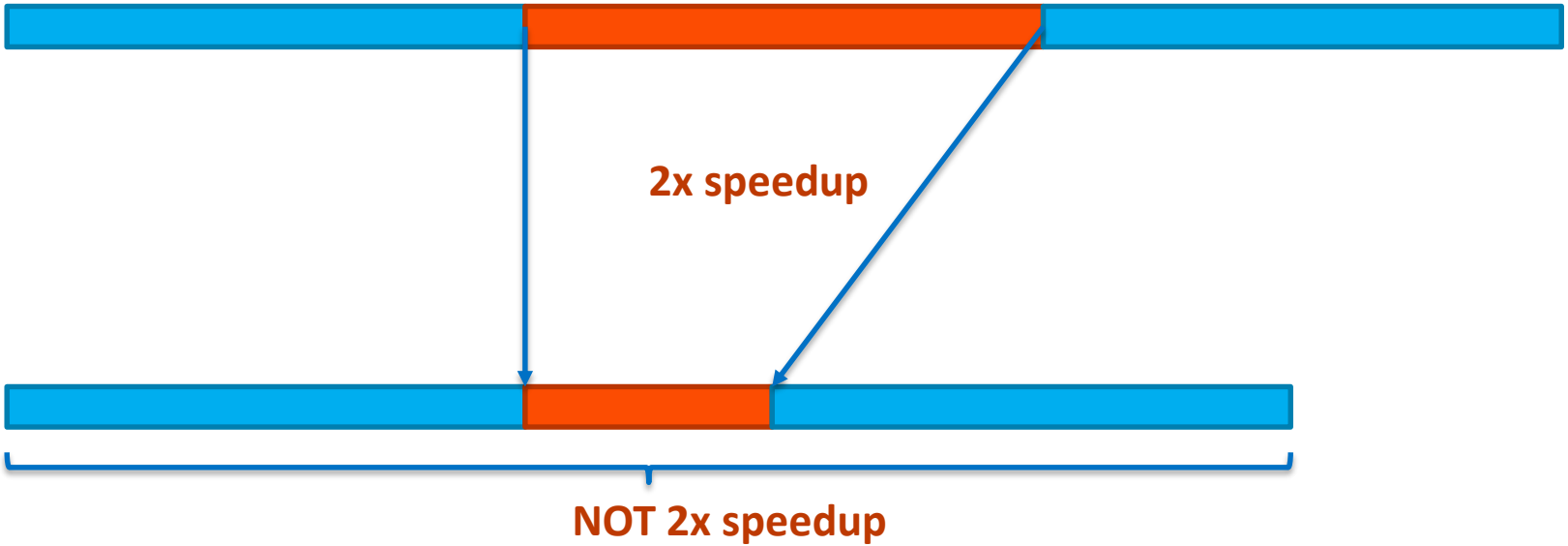
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

AMDHAL'S LAW

- $Stotal = \frac{1}{(1-p)+\frac{p}{s}}$
- S = speedup
- P = proportion of execution time that benefit from threading

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

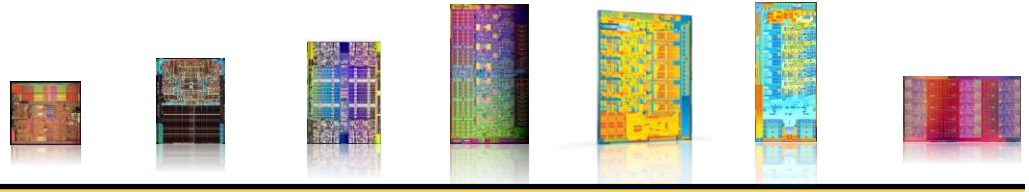
AMDHAL'S LAW



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

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 Skylake EP	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing[1]
Core(s)	1	2	4	6	8	12	28	61	60+
Threads	2	2	8	12	16	24	56	244	
SIMD Width	128	128	128	128	256	256	256	512	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

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

DON'T USE A SINGLE VECTOR LANE!

UN-VECTORIZED AND UN-THREADED SOFTWARE WILL UNDER PERFORM

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SINGLE INSTRUCTION MULTIPLE DATA (SIMD)

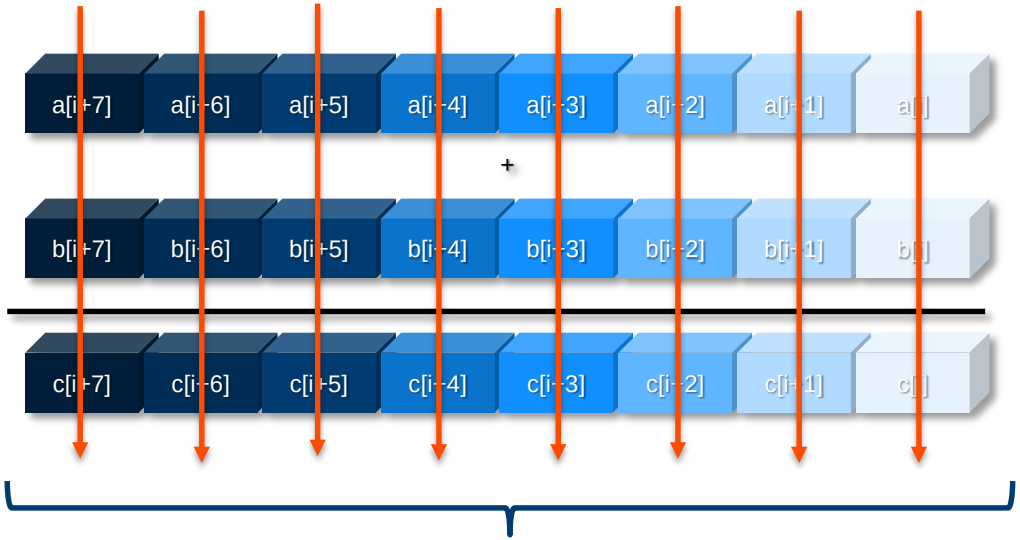
```
for(i = 0; i <= MAX; i++)
   c[i] = a[i] + b[i];
```

```
for(i = 0; i <= MAX; i+8)
   c[i:8] = a[i:8] + b[i:8];
```

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.
intel Software

VECTORIZATION TERMS

Vector lanes



Vector length(VL): Elements of the vector

All elements of vector are of the same data types

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

VECTORIZATION OF A LOOP

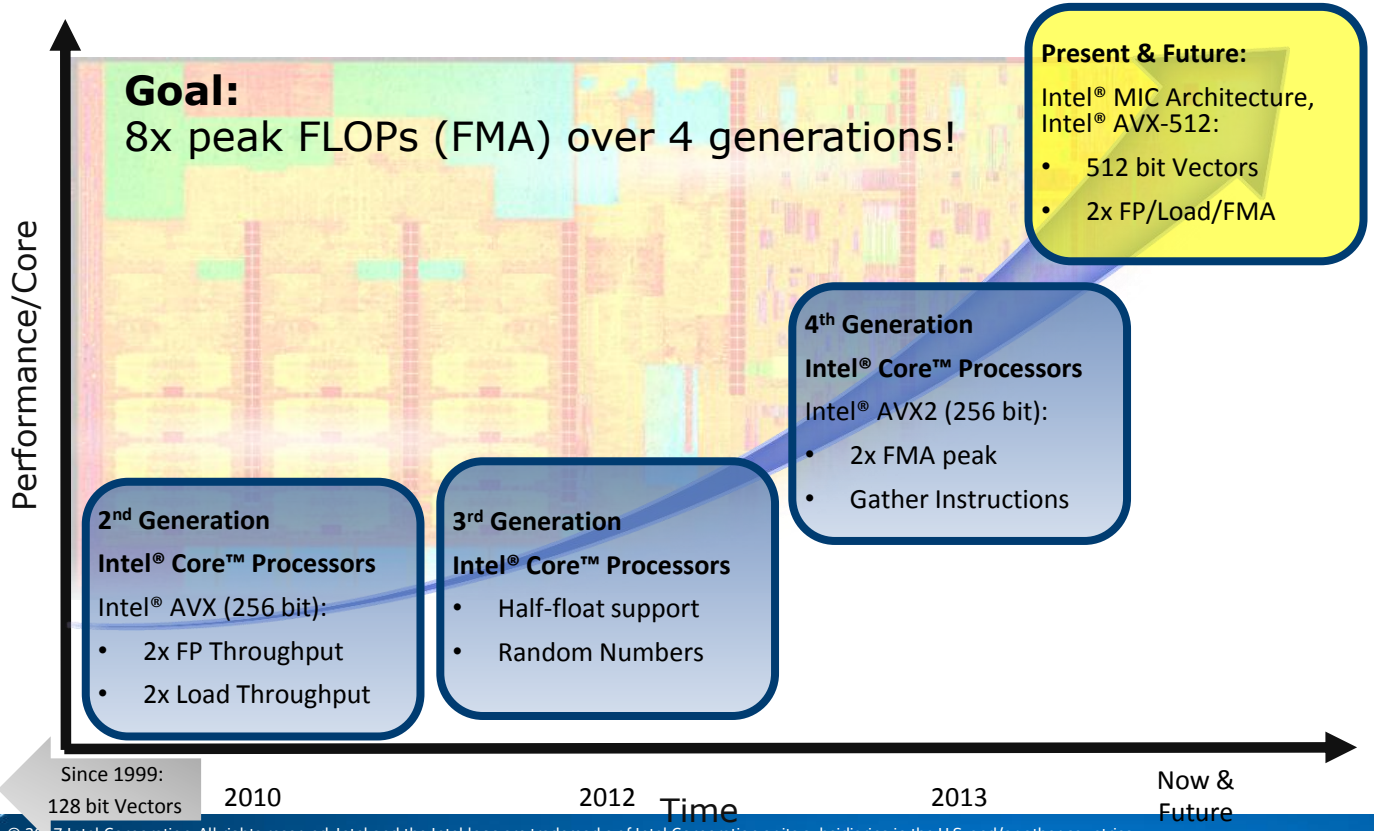
```
for(i = 0; i < MAX; i++)
    A[i] = B[i] + C[i];
```

```
// Vectorized loop body
  for(i = 0; i < MAX; i+=VL)
      A[i:VL] = B[i:VL] + C[i:VL];

// loop reminder
  for(i = i-VL; i < MAX; i++)
      A[i] = B[i] + C[i];
```

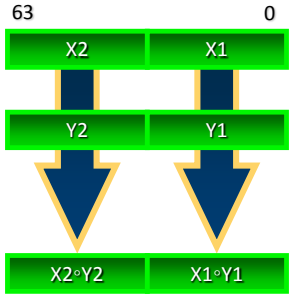
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

EVOLUTION OF SIMD FOR INTEL PROCESSORS



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SIMD TYPES FOR INTEL® ARCHITECTURE I



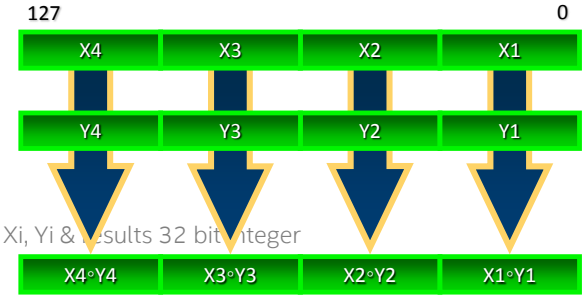
MMX™

Vector size: **64 bit**

Data types:

- 8, 16 and 32 bit integer

VL: 2, 4, 8



SSE

Vector size: **128 bit**

Data types:

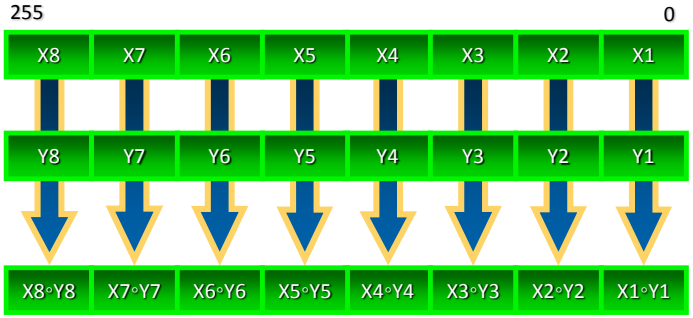
- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

VL: 2, 4, 8, 16

- Illustrations: Xi, Yi & results 32 bit integer

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SIMD TYPES FOR INTEL® ARCHITECTURE II



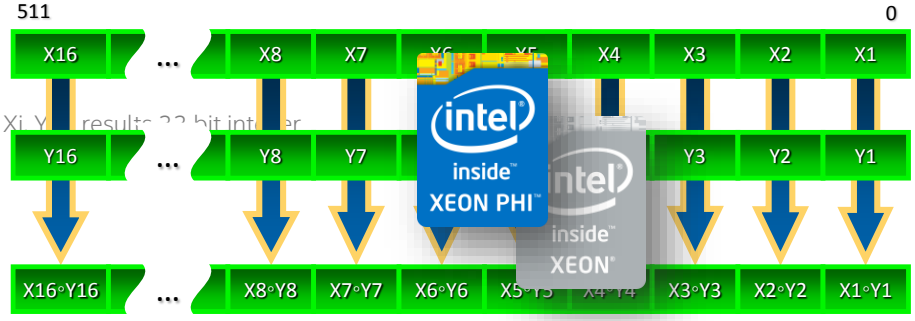
AVX

Vector size: **256 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

VL: 4, 8, 16, 32



Intel® AVX-512 & Intel® MIC Architecture

Vector size: **512 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

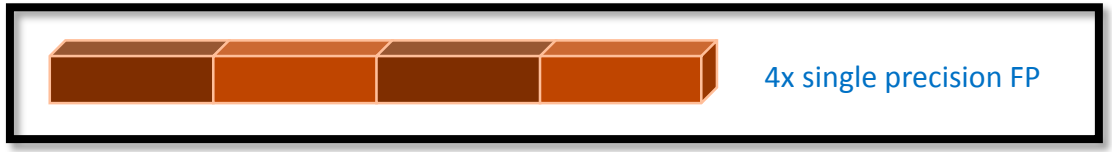
VL: 8, 16, 32, 64

- Illustrations: Xi Y… results 32 bit int… er

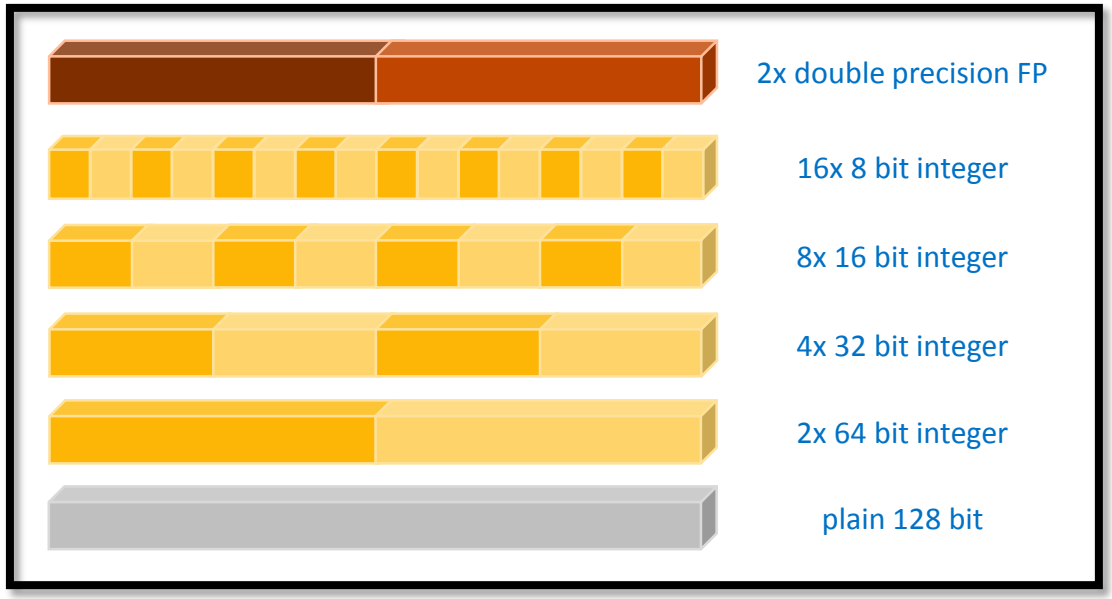
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SSE VECTOR TYPES

Intel® SSE

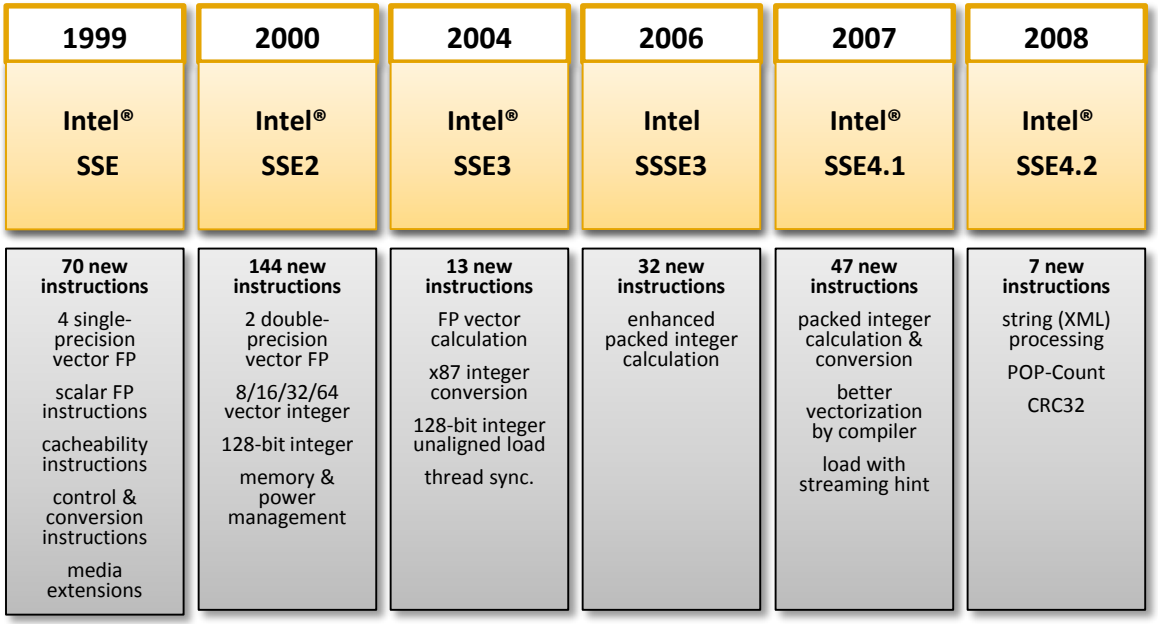


Intel® SSE2



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SSE GENERATIONS



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

SSE PACKED VS. SCALAR

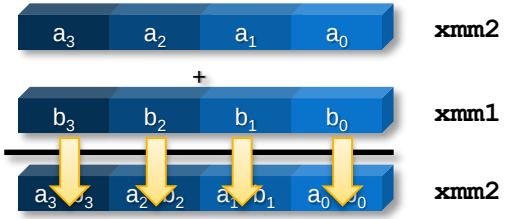
- **Packed** SSE instructions operate on all elements per vector
- Most of these instructions have **scalar** versions operating only on one element of vector
- Avoid scalar versions and only **use packed instructions** to exploit SIMD capabilities!

Packed single-precision FP Addition:

```
addps xmm2, xmm1
```

single-precision FP data type

packed execution mode

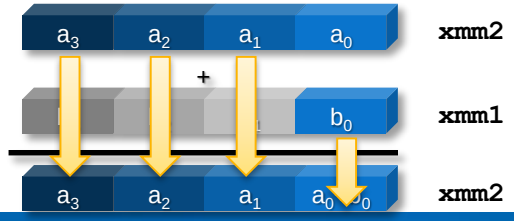


Scalar single-precision FP Addition:

```
addss xmm2, xmm1
```

single-precision FP data type

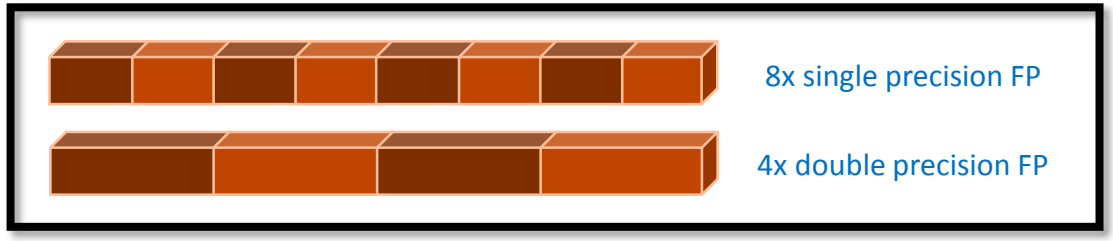
scalar execution mode



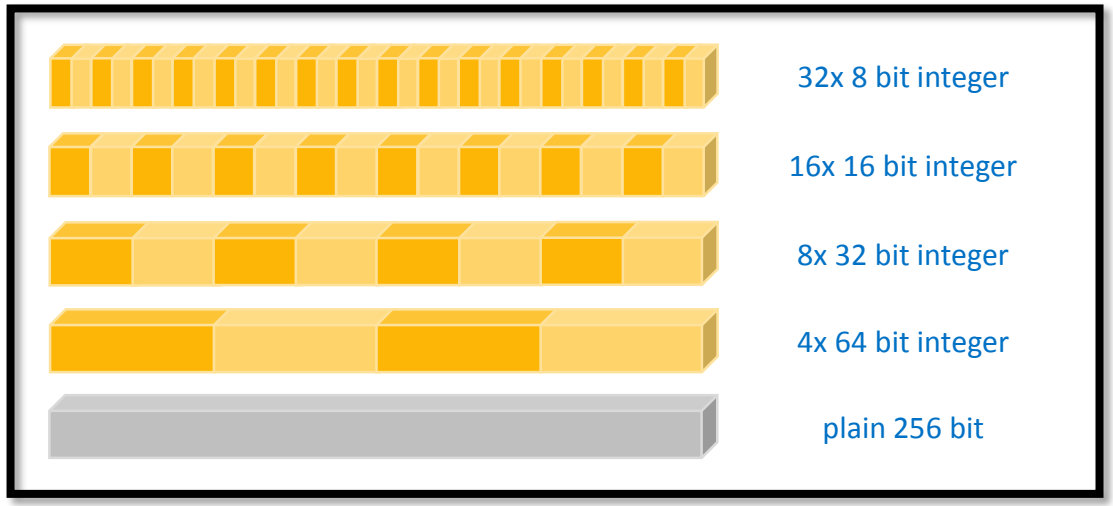
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

AVX VECTOR TYPES

Intel® AVX



Intel® AVX2



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

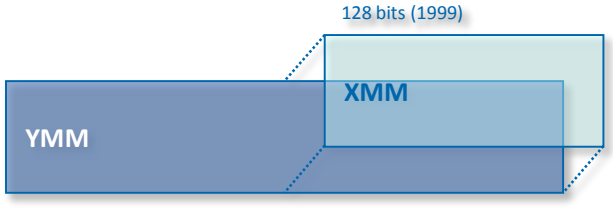
AVX GENERATIONS

- 2010: Initial version of **Intel® AVX** in 2nd generation Intel® Core™ processors:
  - Double register size of SSE, twice as much vector elements (2x peak FLOP)
  - Support for single- & double-precision FP
  - Load/Store size increased from 128 bit to 256 bit!
- 2012: 3rd generation Intel® Core™ processor improvements:
  - Non-deterministic random number generator
  - Half-precision conversion (from/to single-precision)
- 2013: 4th generation Intel® Core™ processor improvements:
  - **Intel® AVX2** (with integer support)
  - FMA (2x more peak FLOP with same vector length)
  - Gather non adjacent memory locations
- Future: **Intel® AVX-512**

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

AVX IS SSE EXTENSION

- A 256 bit vector extension to SSE:
  - SSE uses dedicated 128 bit registers called XMM (8 for IA-32 & 16 for Intel® 64)
  - Extends all XMM registers to 256 bit called YMM
  - Lower 128 bit of YMM register are mapped/shared with XMM
- AVX works on either
  - The whole 256 bit
  - The lower 128 bit; zeros the higher 128 bit
  - AVX counterparts for almost all existing SSE instructions:
    For initial generation (Intel® AVX) full 256 bit vectors for FP only; integers will follow!



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

INTEL® ADVISOR

INTEL® ADVISOR

Boosting your application by threading and vectorization

- Compiler will not always vectorize
  - Check for Loop Carried Dependencies using Intel® Advisor
  - All clear? Force vectorization.
    C++ use: pragma simd, Fortran use: SIMD directive
- Not all vectorization is efficient vectorization
  - Stride of 1 is more cache efficient than stride of 2 and greater. Analyze with Intel® Advisor.
  - Consider data layout changes
    Intel® SIMD Data Layout Templates can help

Arrays of structures are great for intuitively organizing data, but are much less efficient than structures of arrays. Use the Intel® SIMD Data Layout Templates (Intel® SDLT) to map data into a more efficient layout for vectorization.

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

GET FASTER CODE FASTER! INTEL® ADVISOR

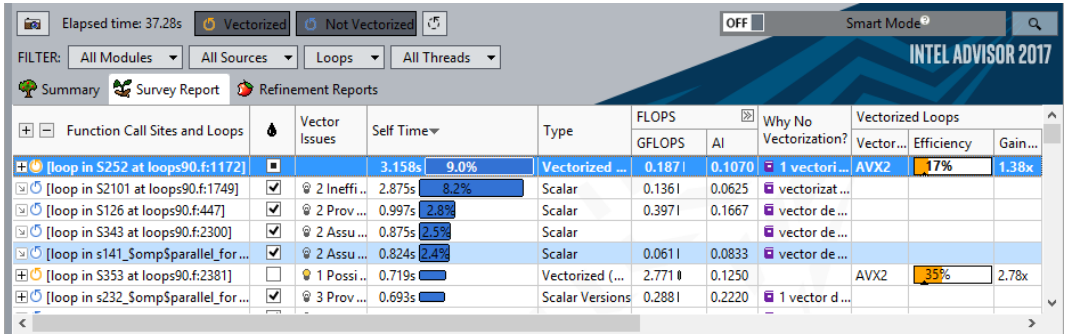
Vectorization Optimization

Have you:

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

Data Driven Vectorization:

- What vectorization will pay off most?
- What's blocking vectorization? Why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?



"Intel® Advisor's Vectorization Advisor permitted me to focus my work where it really mattered. When you have only a limited amount of time to spend on optimization, it is invaluable."

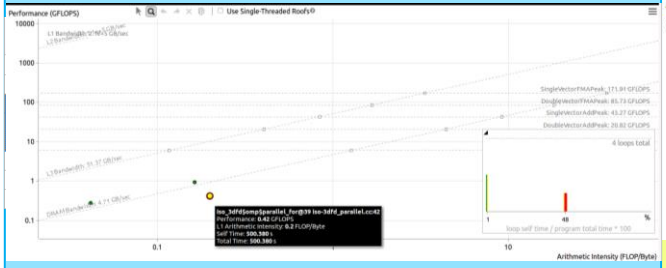
Gilles Civario
Senior Software Architect
Irish Centre for High-End Computing

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

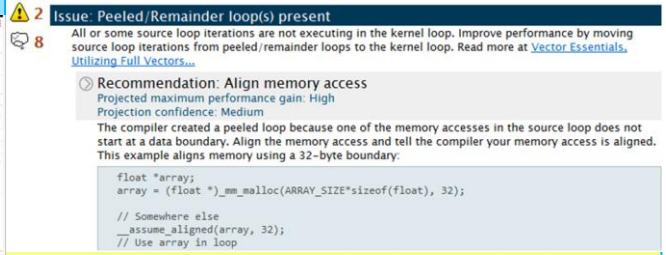
4 STEPS TO EFFICIENT VECTORIZATION

Intel® Advisor – Vectorization Advisor

1. Compiler diagnostics + Performance Data + Roofline Model + SIMD efficiency information



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



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

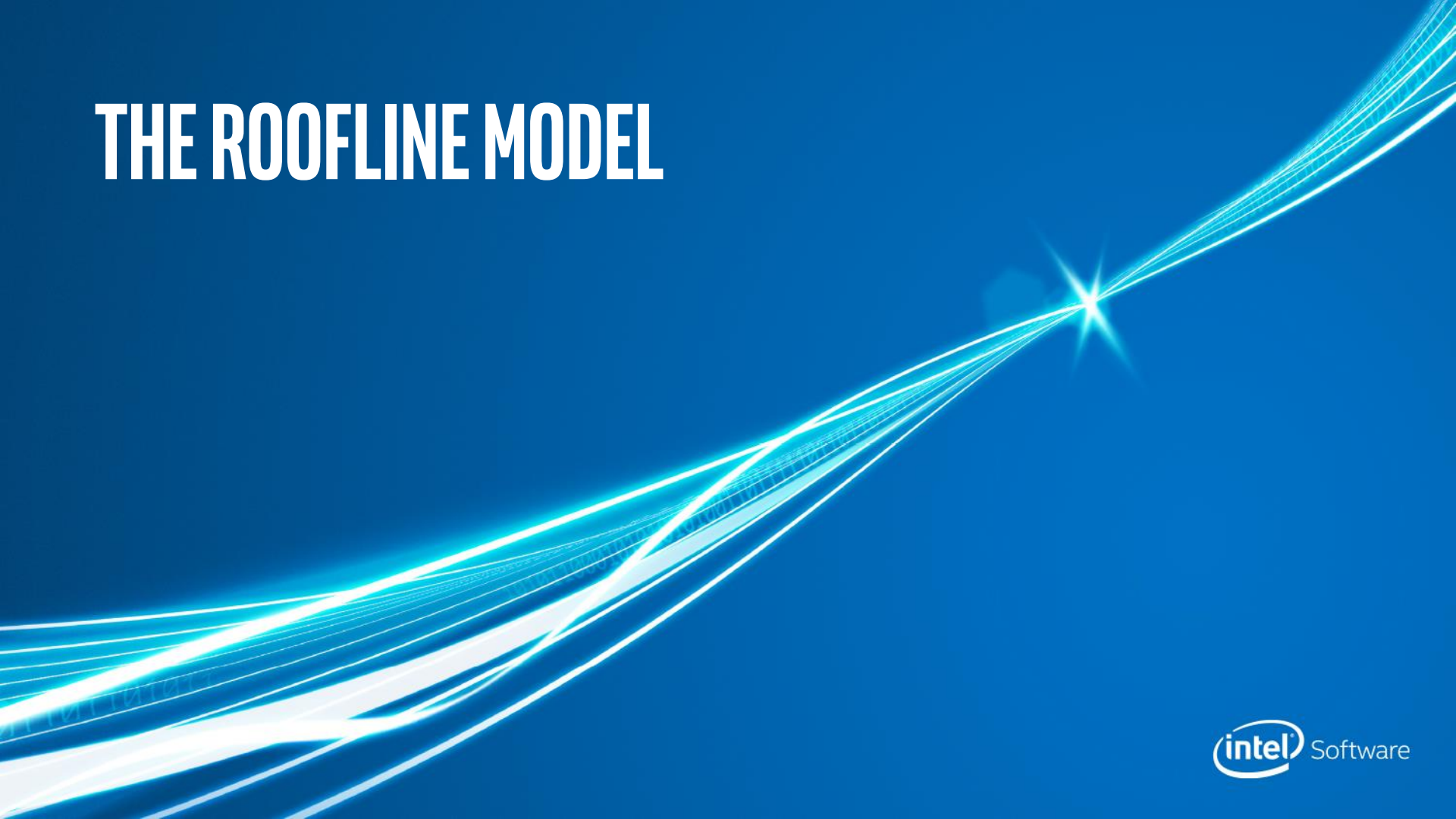
ID		Stride	Type	Source	Modules	Alignment
P22		0; 0; 1	Unit stride	runCRawLoops.cxx:637	lcals.exe	
P23		0; 0	Unit stride	runCRawLoops.cxx:638	lcals.exe	
P30		-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cxx:628	lcals.exe	

```
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];
```

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

THE ROOFLINE MODEL



WHAT IS THE ROOFLINE MODEL ?

Do you know how fast you should run ?

- Comes from Berkeley
- Performance is limited by equations/implementation & code generation/hardware
- 2 hardware limitations
  - PEAK Flops
  - PEAK Bandwidth
- The application performance is bounded by hardware specifications

$$\text{Gflop/s} = \min \begin{cases} \textbf{Platform PEAK} \\ \textbf{Platform BW} * \textbf{AI} \end{cases}$$

Arithmetic Intensity (Flops/Bytes)

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

PLATFORM PEAK FLOPS

How many floating point operations per second

$$\text{Gflop/s} = min \begin{cases} Platform\ PEAK \\ Platform\ BW * AI \end{cases}$$

- Theoretical value can be computed by specification

  Example with 2 sockets Intel® Xeon® Processor E5-2697 v2

  PEAK FLOP = 2 x 2.7 x 12 x 8 x 2 = **1036.8 Gflop/s**

  - Number of sockets
  - Core Frequency
  - Number of cores
  - Number of single precision element in a SIMD register
  - 1 port for addition, 1 for multiplication

- More realistic value can be obtained by running **Linpack**

  =~ 930 Gflop/s on a 2 sockets Intel® Xeon® Processor E5-2697 v2

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

PLATFORM PEAK BANDWIDTH

How many bytes can be transferred per second

$$\text{Gflop/s} = min \begin{cases} Platform\ PEAK \\ Platform\ BW * AI \end{cases}$$

- Theoretical value can be computed by specification

  Example with 2 sockets Intel® Xeon® Processor E5-2697 v2

  PEAK BW = 2 x 1.866 x 8 x 4 = 119 GB/s

  Number of sockets

  Memory Frequency

  Byte per channel

  Number of mem channels

- More realistic value can be obtained by running **Stream**

  =~ 100 GB/s on a 2 sockets Intel® Xeon® Processor E5-2697 v2

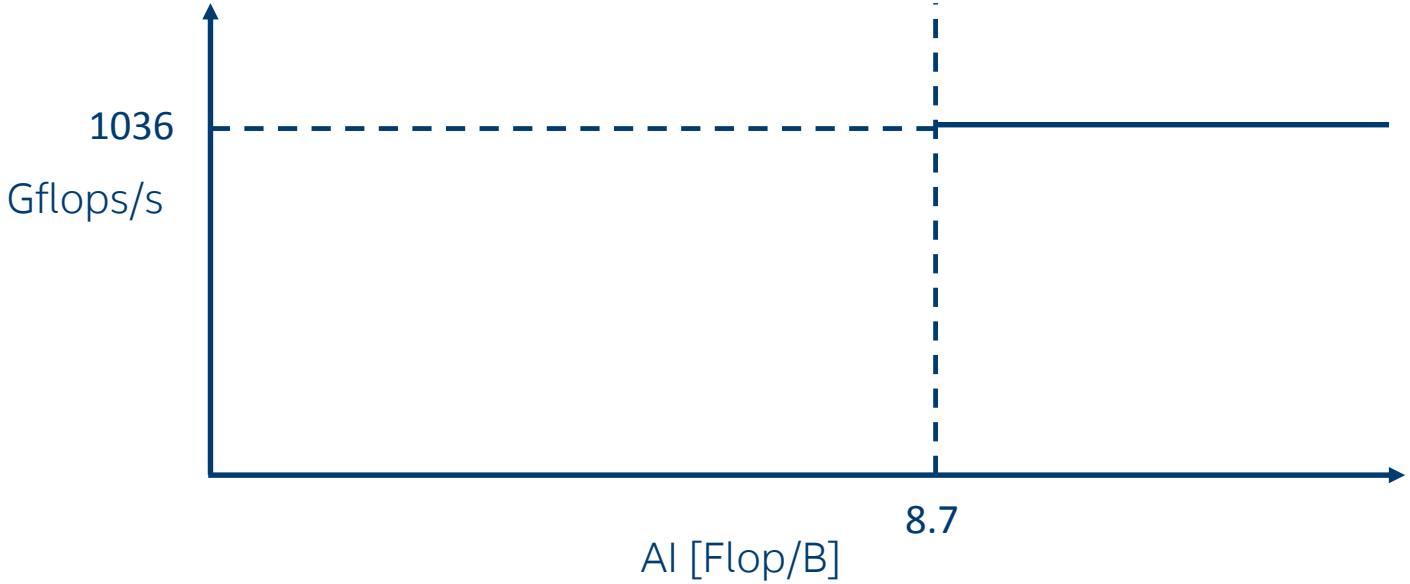
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

DRAWING THE ROOFLINE

Defining the speed of light

$$\text{Gflop/s} = \min \begin{cases} \textit{Platform PEAK} \\ \textit{Platform BW} * \textit{AI} \end{cases}$$

2 sockets Intel® Xeon® Processor E5-2697 v2
Peak Flop = 1036 Gflop/s
Peak BW = 119 GB/s



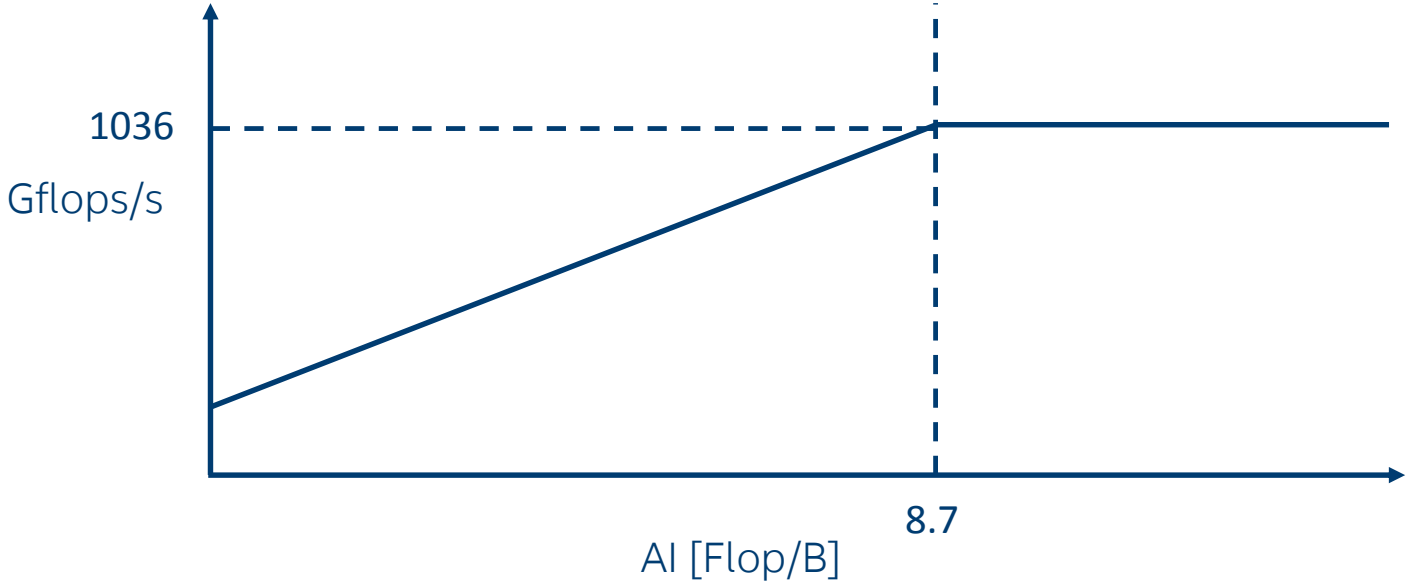
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

DRAWING THE ROOFLINE

Defining the speed of light

$$\text{Gflop/s} = \min \begin{cases} \textit{Platform PEAK} \\ \textit{Platform BW} * \textit{AI} \end{cases}$$

2 sockets Intel® Xeon® Processor E5-2697 v2
Peak Flop = 1036 Gflop/s
Peak BW = 119 GB/s



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

WHAT IS THE PERFORMANCE BOUNDARY?

Manual way to do it

- Manual counting on matrix/matrix multiplication

```
for(i=0; i<N; i++)
    for(j=0; j<N; j++)
        for(k=0; k<N; k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j]
```

- # add = N * N * N #Read = 3 * N * N * 4 bytes

  # mul = N * N * N #Write = N * N * 4 bytes

- $AI = \frac{2N^3}{16N^2} = \frac{1}{8}N$

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

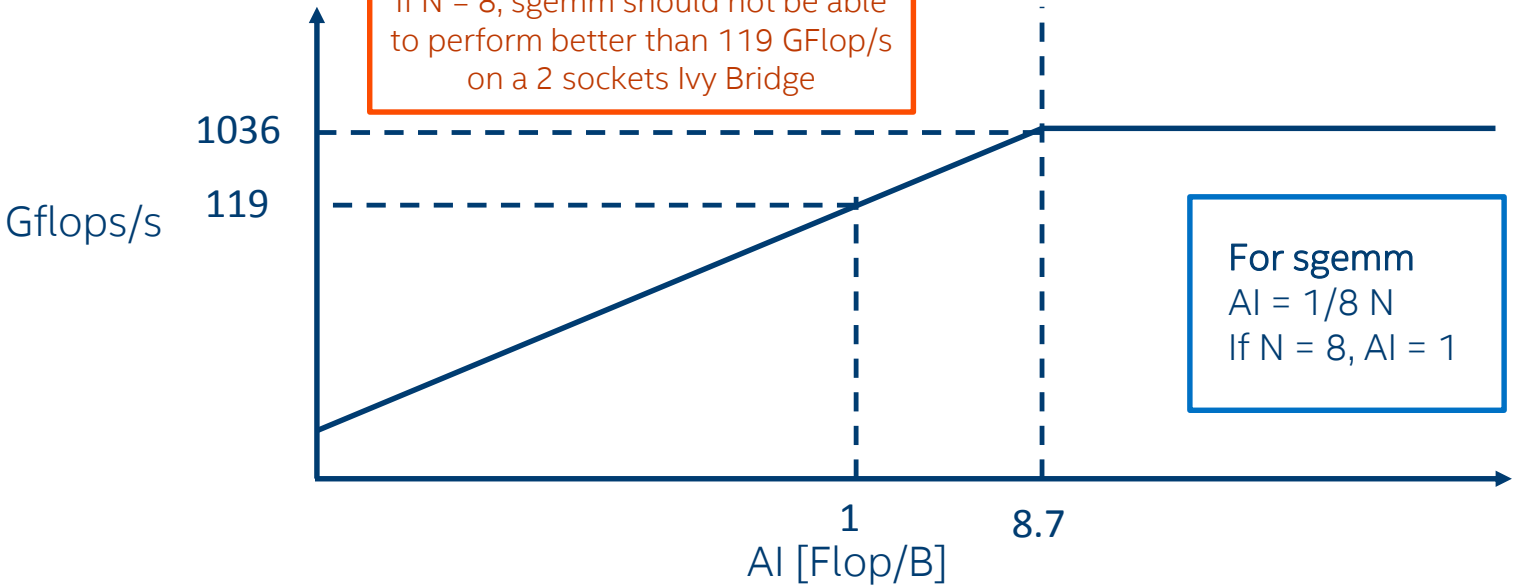
COMPUTE THE MAXIMUM PERFORMANCE

BW * Arithmetic Intensity

$$\text{Gflop/s} = \min \begin{cases} \textbf{Platform PEAK} \\ \textbf{Platform BW} * \textbf{AI} \end{cases}$$

2 sockets Intel® Xeon® Processor E5-2697 v2
Peak Flop = 1036 Gflop/s
Peak BW = 119 GB/s

If N = 8, sgemm should not be able to perform better than 119 GFlop/s on a 2 sockets Ivy Bridge



For sgemm
AI = 1/8 N
If N = 8, AI = 1

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

HOW CAN I DRAW A POINT?

Retrieving real AI and real Flop/s

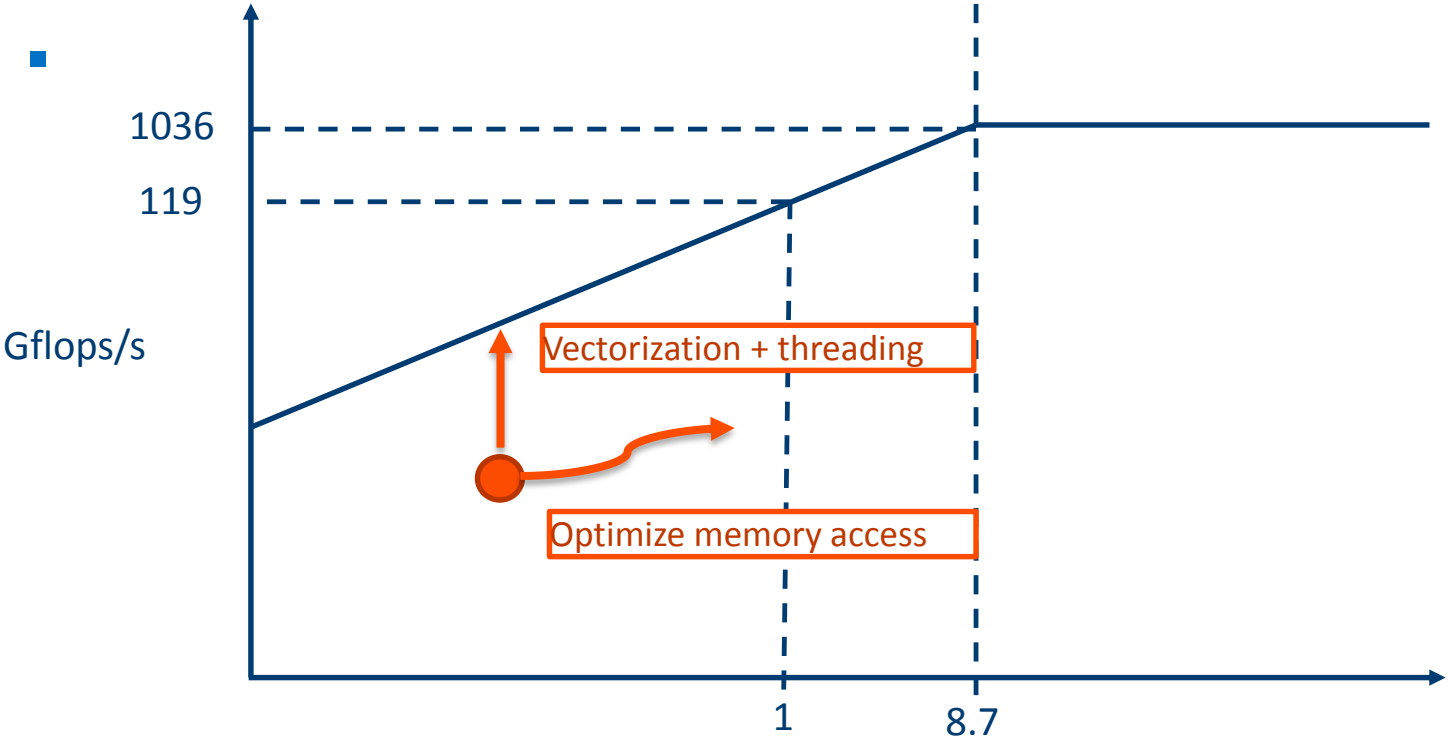
In order to know where your application stands on this graph, you need to extract some metrics

- **Effective flops** can be extracted with SDE, or Hardware counters (if available). In practice, effective flops and theoretical flops (manual counting) should be very close. Adding timers in your application allows to also get flop/s
- **Effective AI** requires the amount of bytes transferred from uncore to core + flops. Amount of bytes transferred can be obtained with emon (reading hardware counters).

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

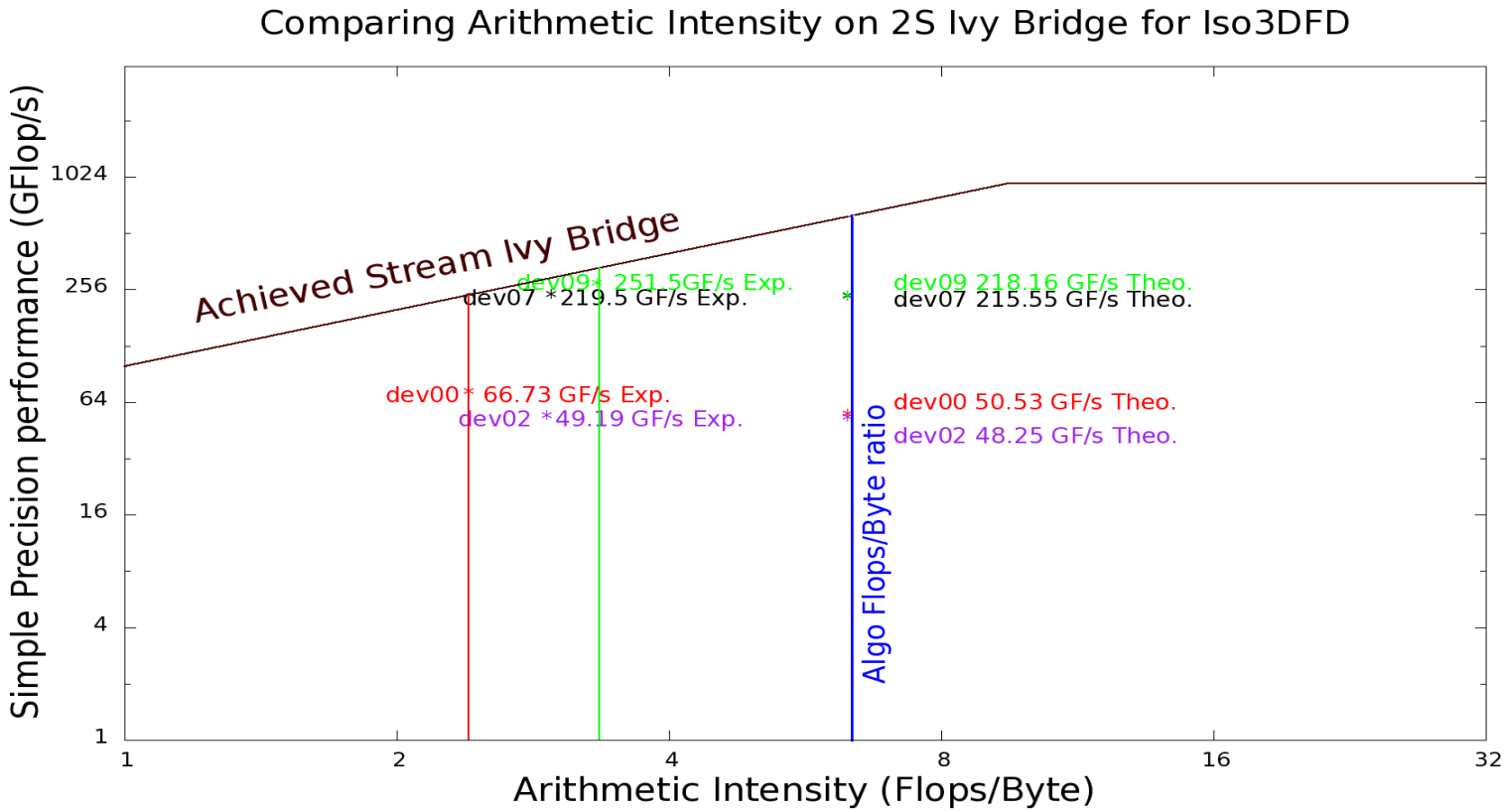
AND NOW?

How to get better performance?



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

UNDERSTANDING THE ROOFLINE



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

ROOFLINE IN INTEL® ADVISOR

The cache aware roofline model

Intel® Advisor implements a Cache Aware Roofline Model (CARM)

- "Algorithmic", "Cumulative (L1+L2+LLC+DRAM)" traffic-based
- Invariant for the given code / platform combination

How does it work ?

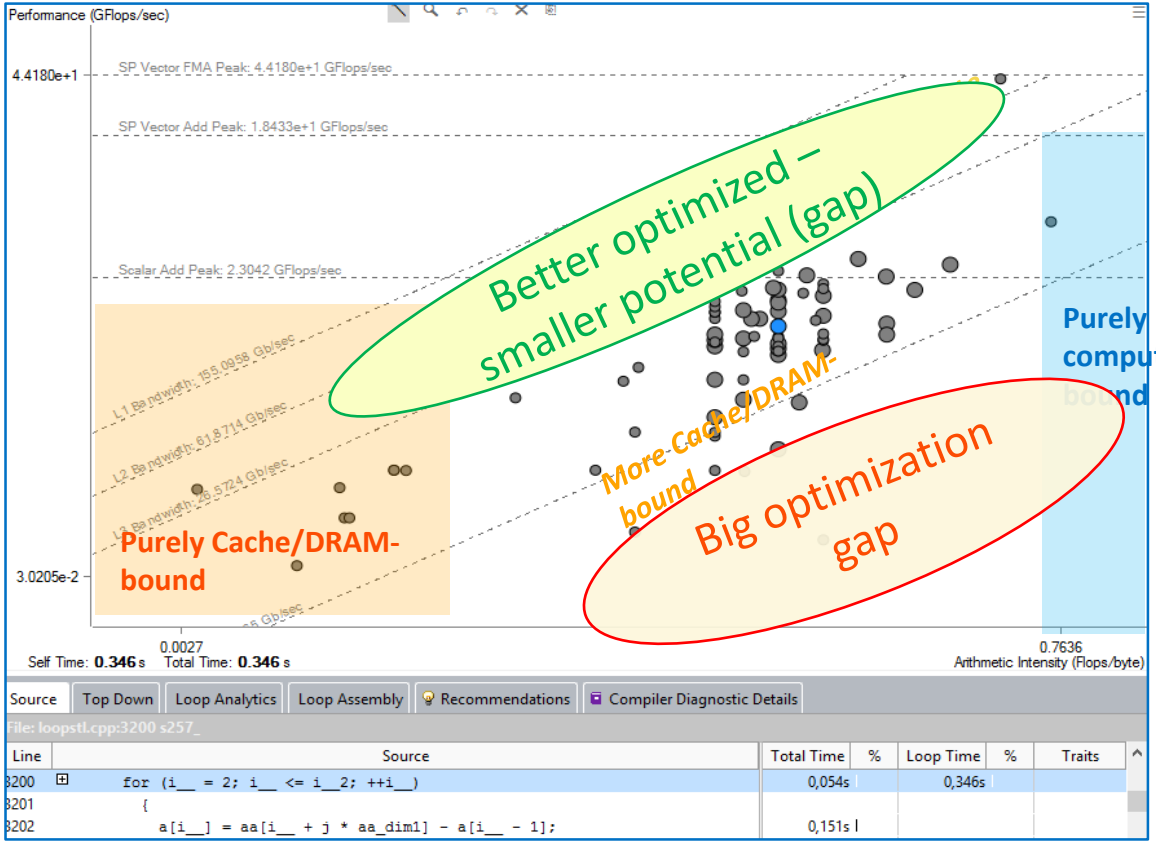
- Counts every memory movement
- Bytes and Flops -> Instrumentation
- Time -> Sampling

CARM: Cache aware Roofline Model
DRAM: DRAM aware Roofline Model
TRAM: Theoretical Roofline Model

Typically AI_CARM < AI_DRAM < AI_TRAM

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

UNDERSTANDING THE ROOFLINE IN INTEL® ADVISOR

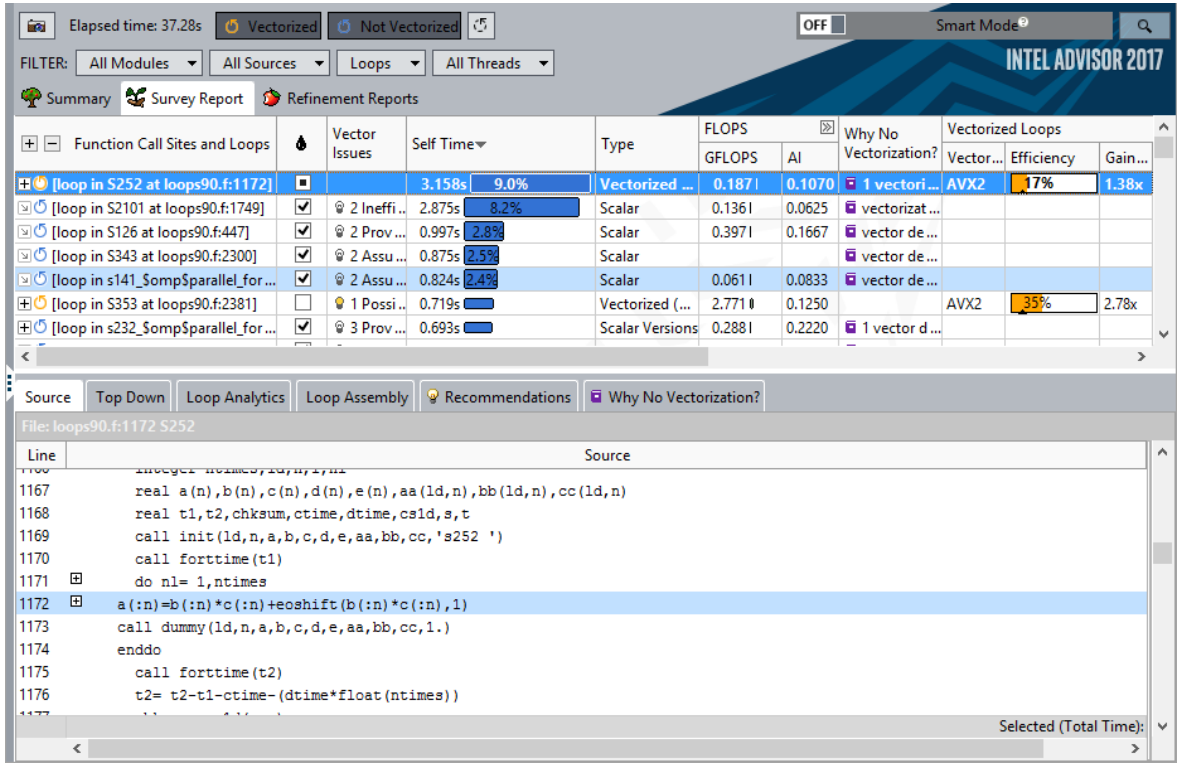


© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

Data Alignment

EFFICIENTLY VECTORIZE YOUR CODE

INTEL ADVISOR – VECTORIZATION ADVISOR



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

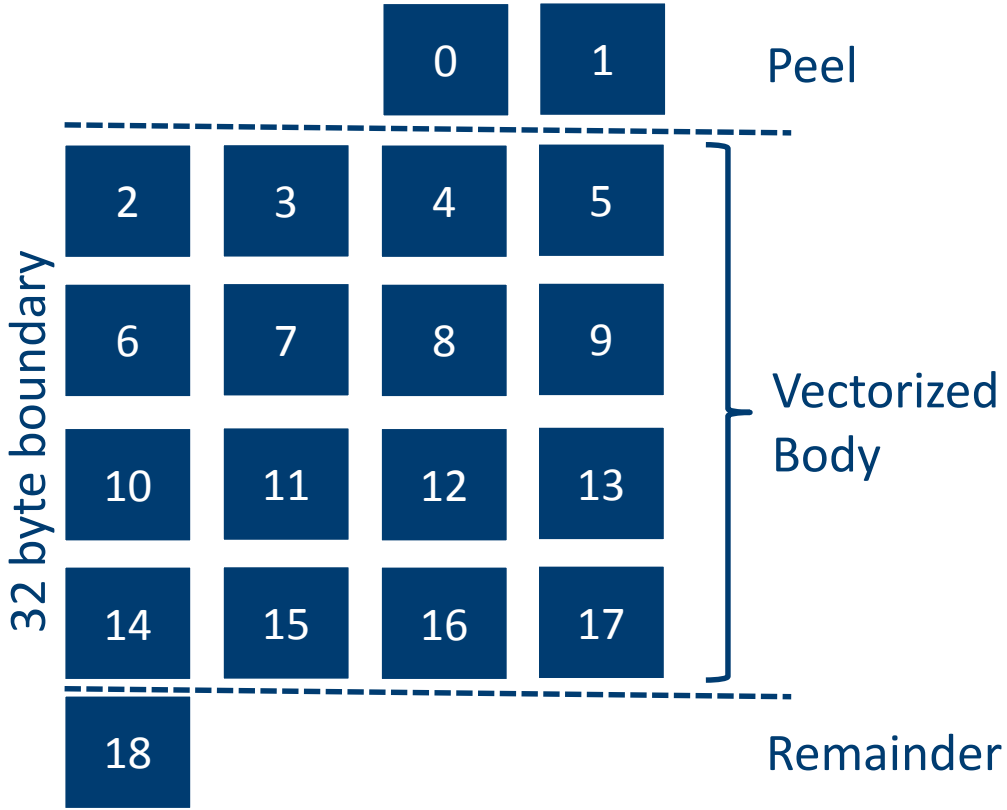
Some Parts of a Vector Are Faster Than Others

This is where we want our loops to be executing!

- 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!

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

WHAT ARE PEELS AND REMAINDERS?

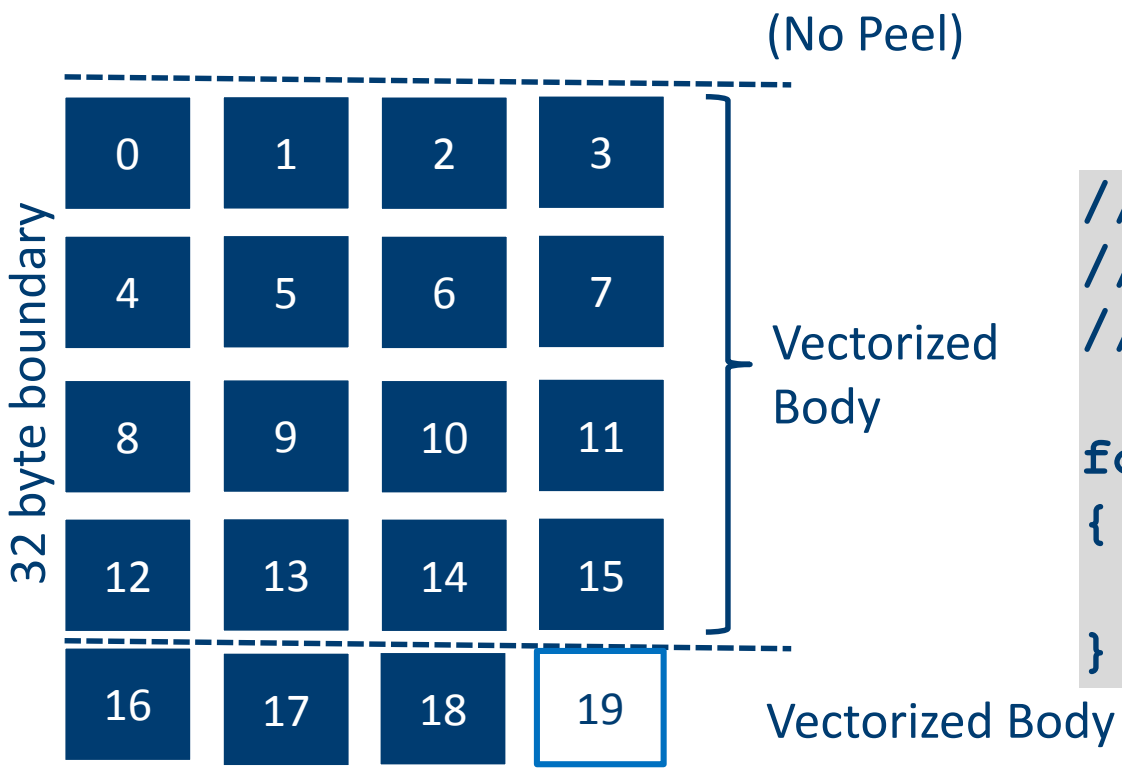



```
// xAVX
// 256 bits wide regs
// holds 4 x 64bit vals

void Func(double *pA)
{
  for (int i=0; i<19;i++)
            pA[i] = …;
}
```

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.
Intel Confidential

WORKING ON ALIGNED ARRAYS

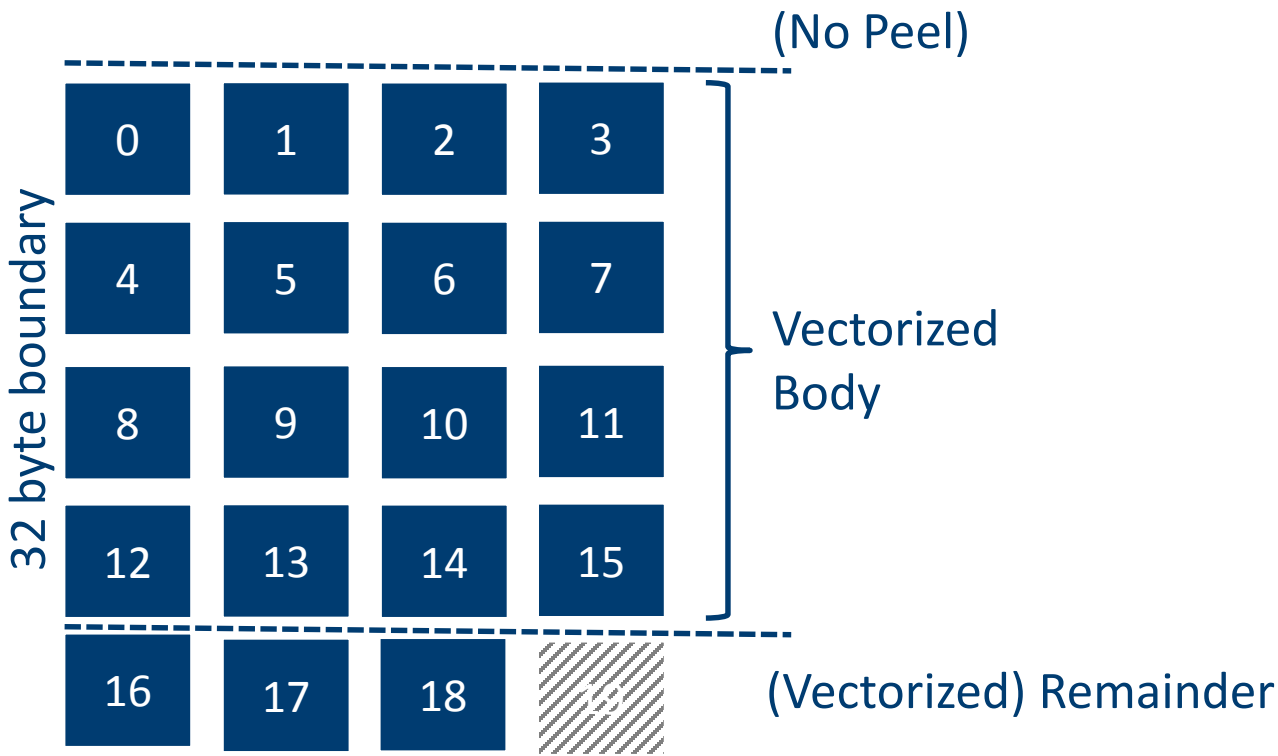



```
// xAVX
// 256 bits wide regs
// holds 4 x 64bit vals

for(i=0; i<lbsy.nq+1; i++)
{
   // . . .
}
```

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

ALIGNMENT AND REMAINDERS ON AVX512

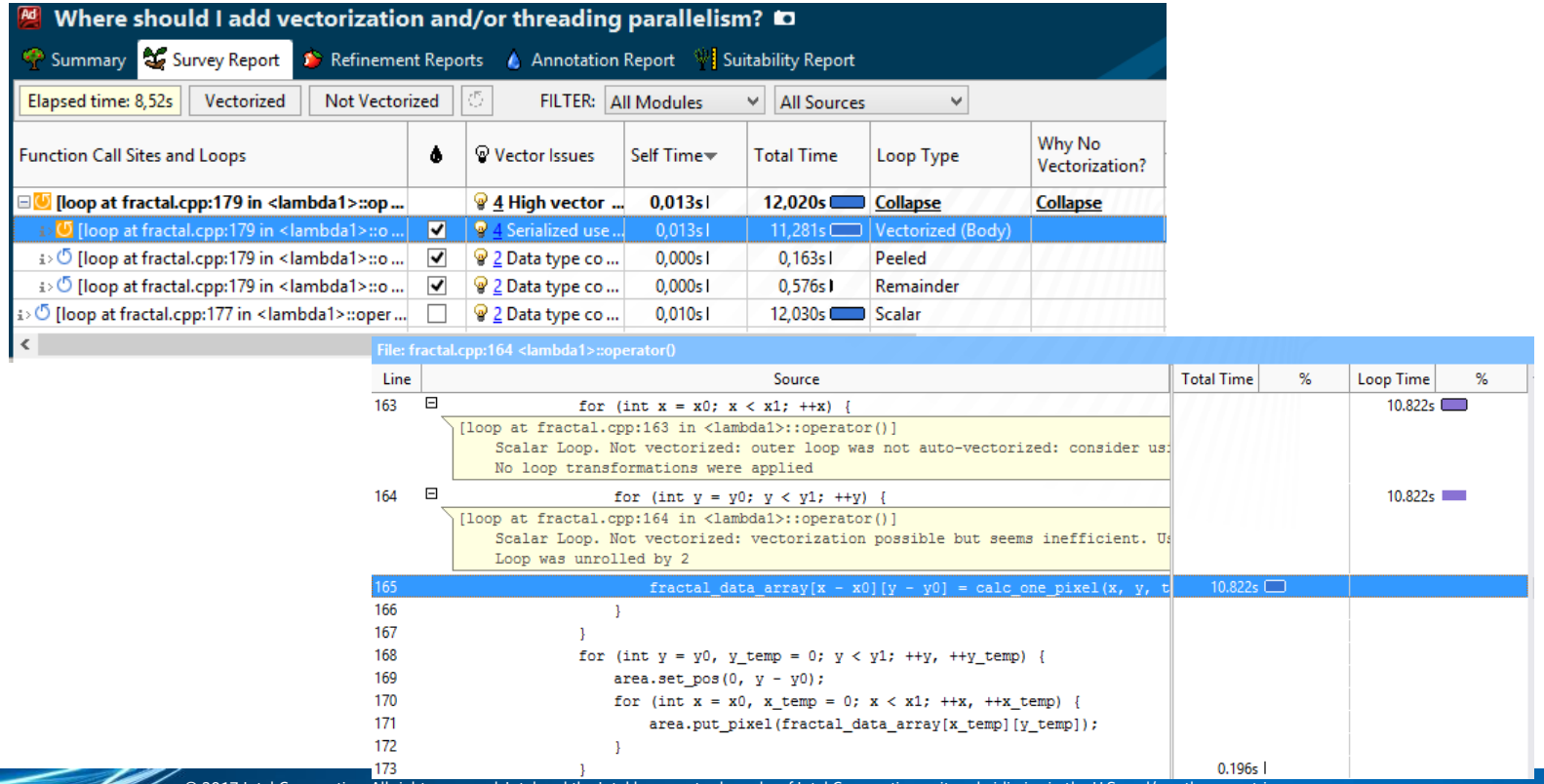


© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

Intel Confidential

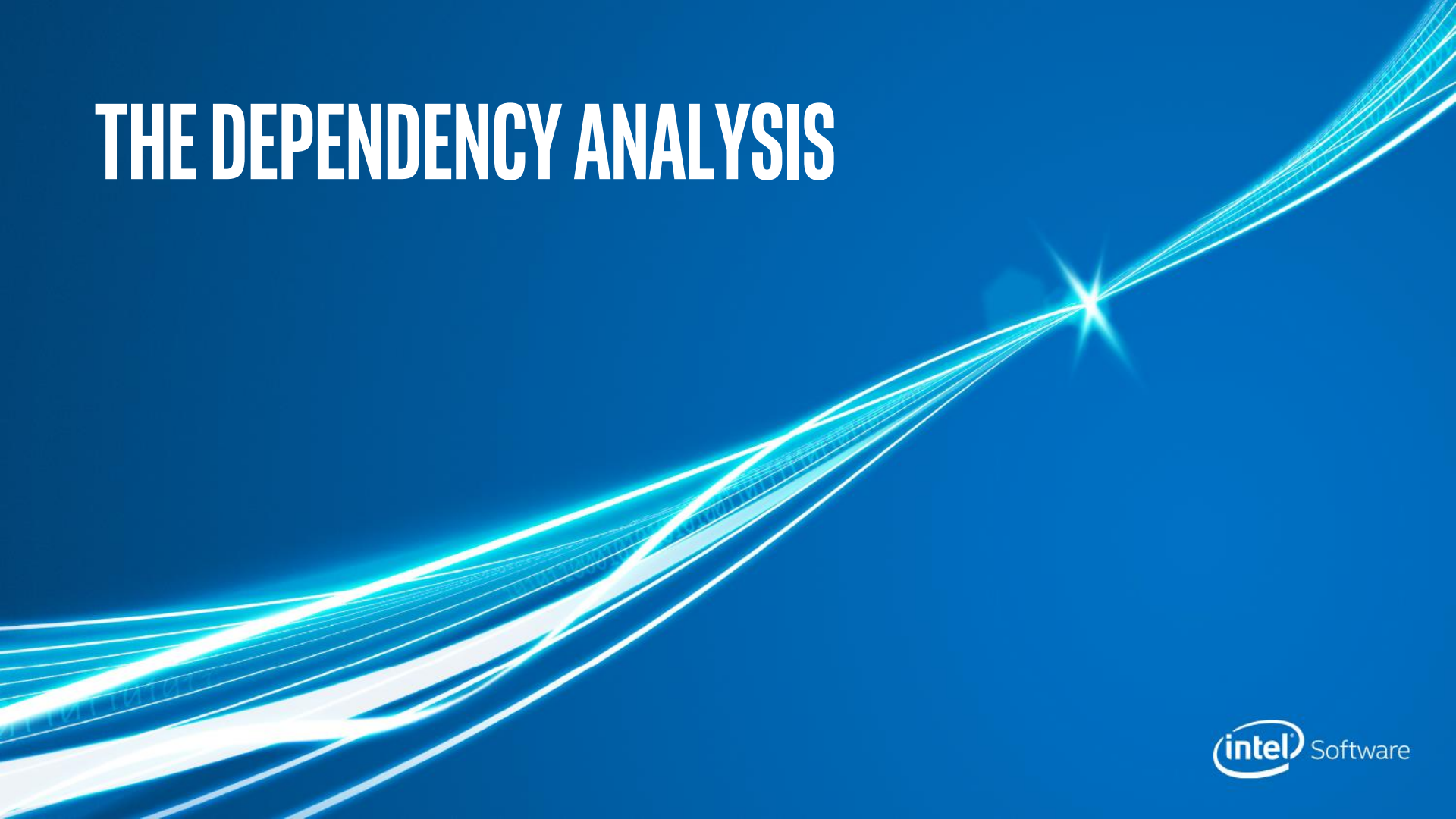
IS MOST EXECUTION IN THE FAST PART OF THE VECTOR?

INTEL ADVISOR SHOWS YOU



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

THE DEPENDENCY ANALYSIS



FORCING VECTORIZATION

Verifying dependency with variables

- Have you seen in compilation reports :
  remark #15344: loop was not vectorized: vector dependence prevents vectorization
- Is it safe to add **#prama omp simd** ?

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)

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M]
```

Input

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

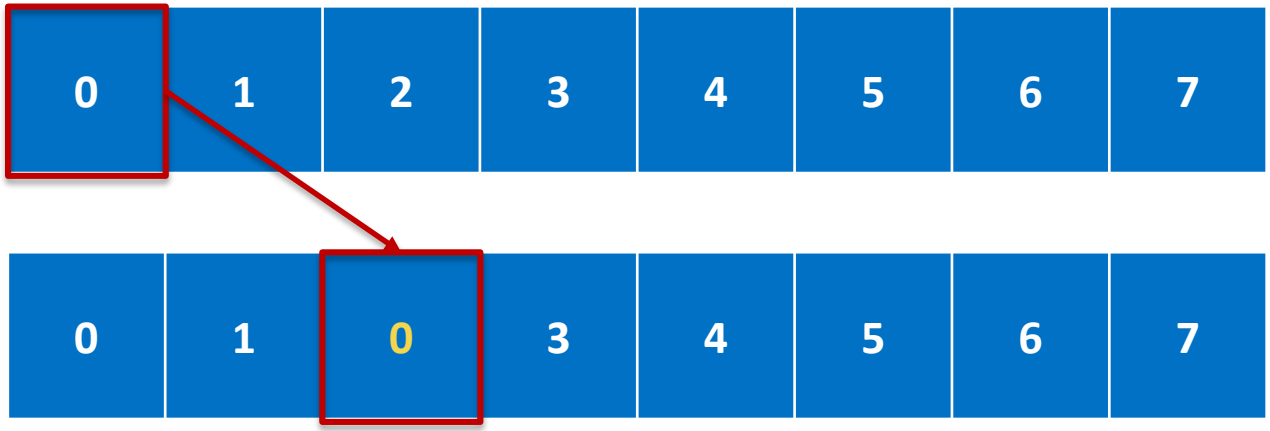
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

i=0



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

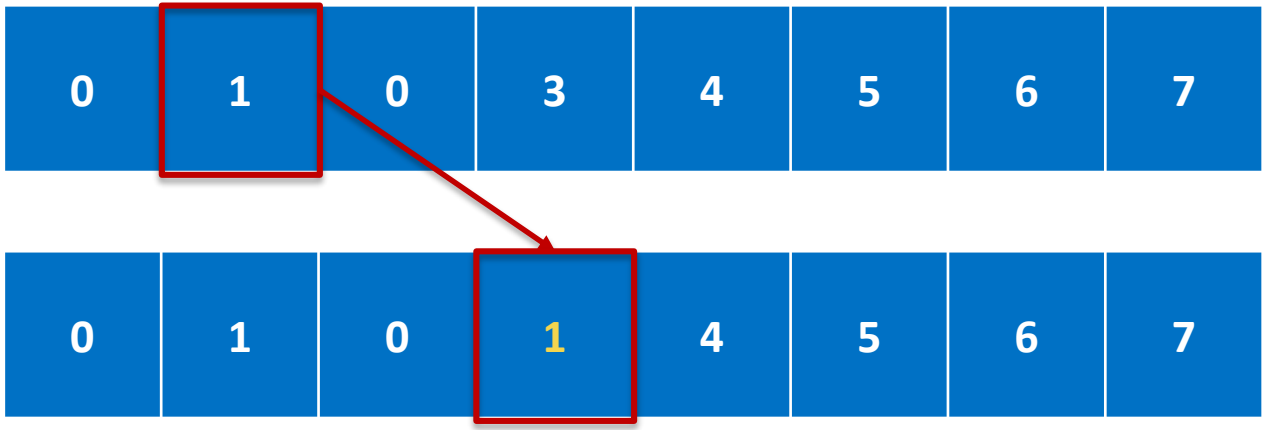
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

i=1



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

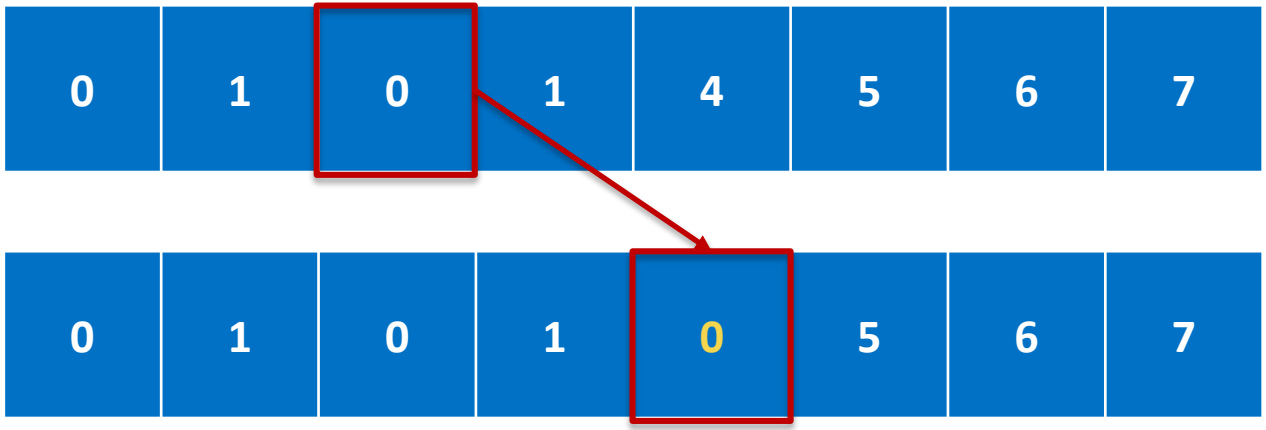
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

i=2



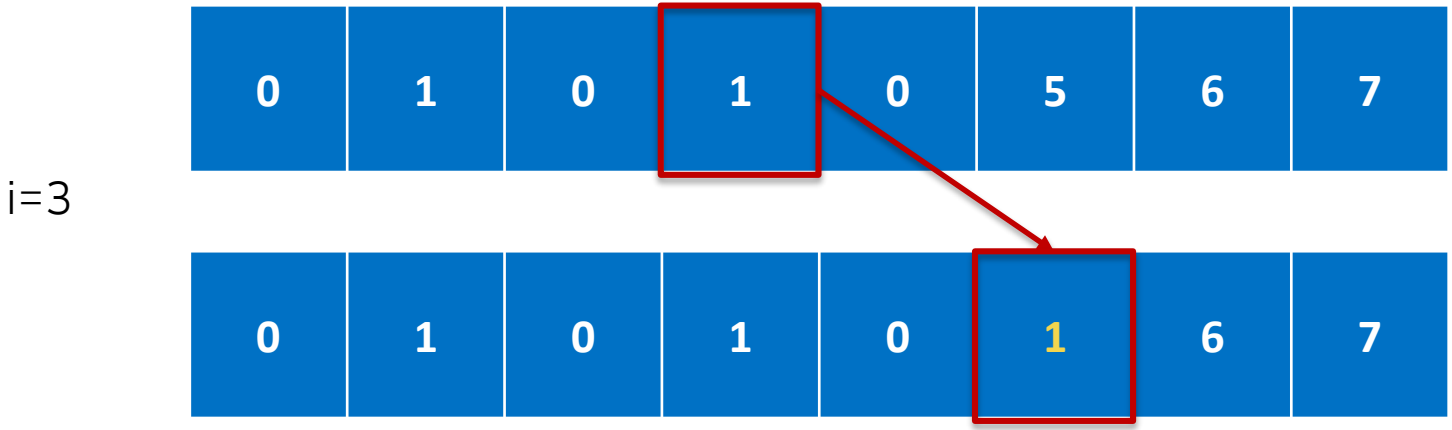
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```



And so on …

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

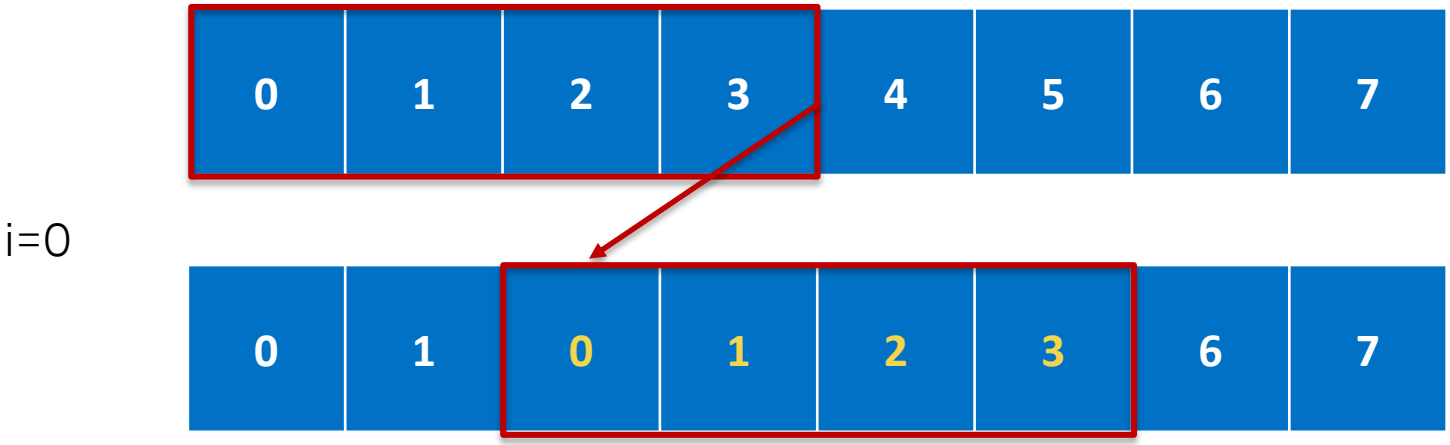
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

With a vector length of 4



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

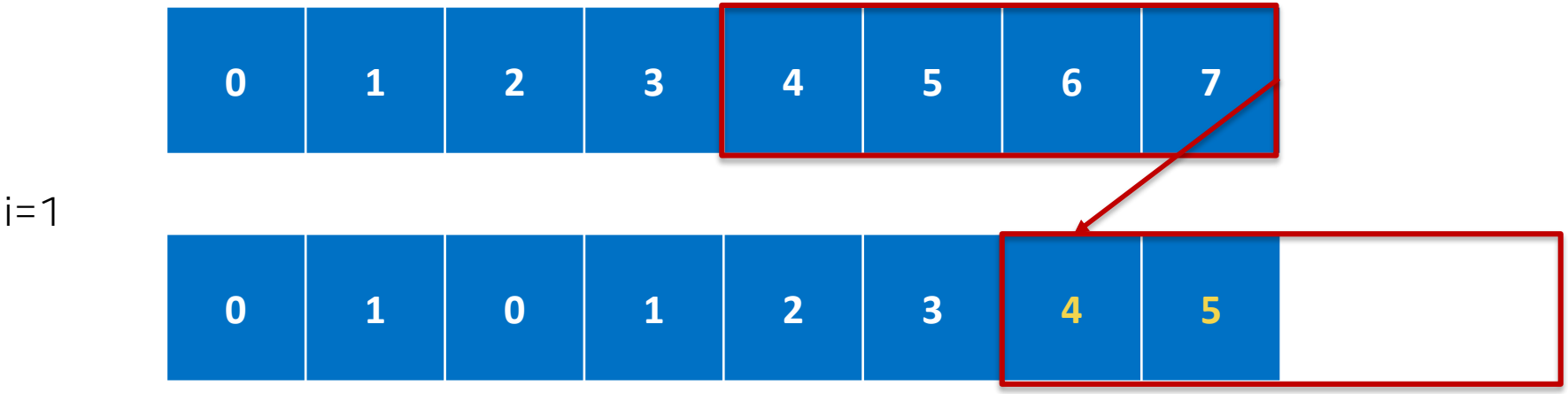
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

With a vector length of 4



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

With a vector length of 4

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

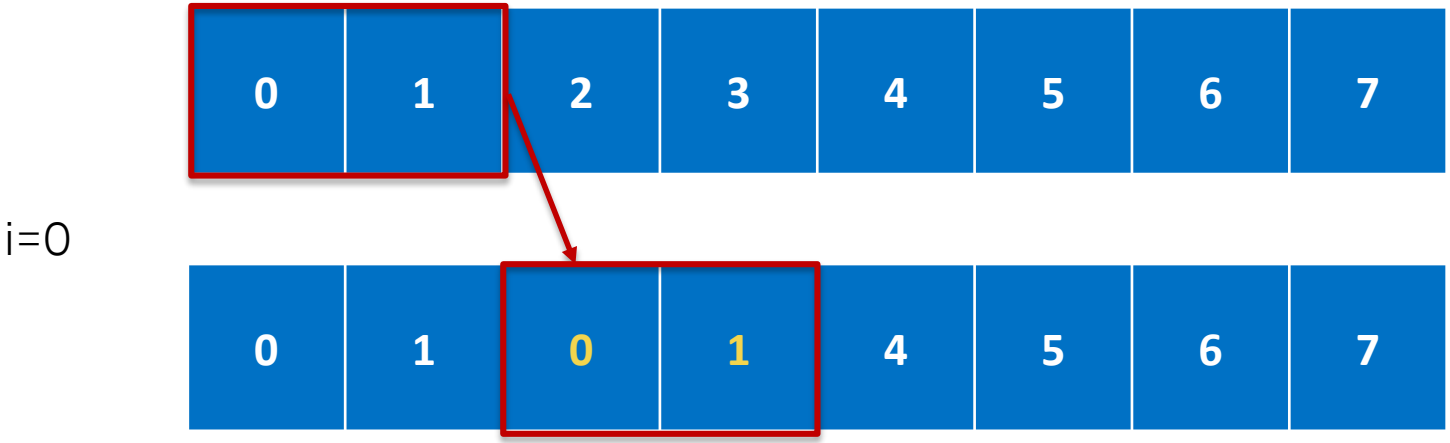
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

With a vector length of 2



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

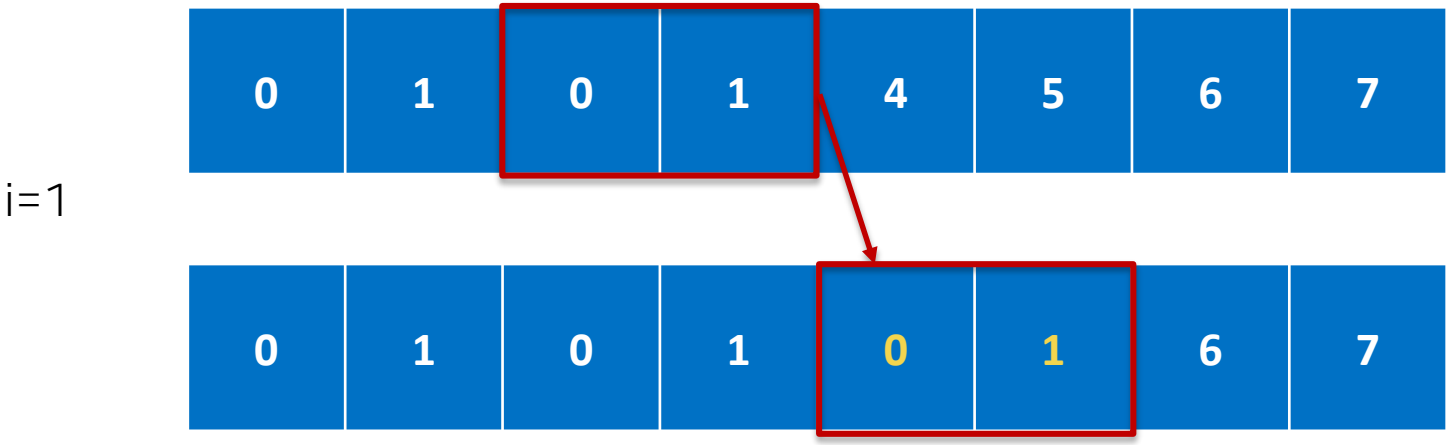
FORCING VECTORIZATION

Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?

```
for (i=0; i<N; i++)
      A[i] = A[i-M] // with M=2
```

With a vector length of 2

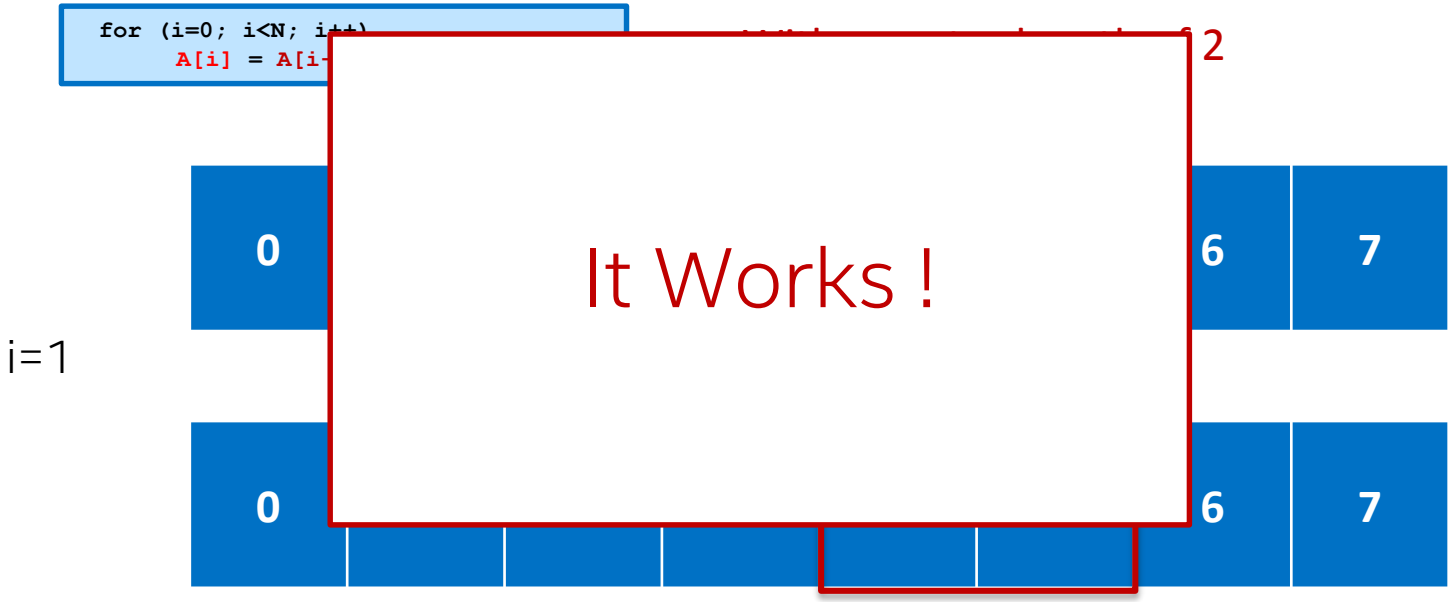


© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

FORCING VECTORIZATION

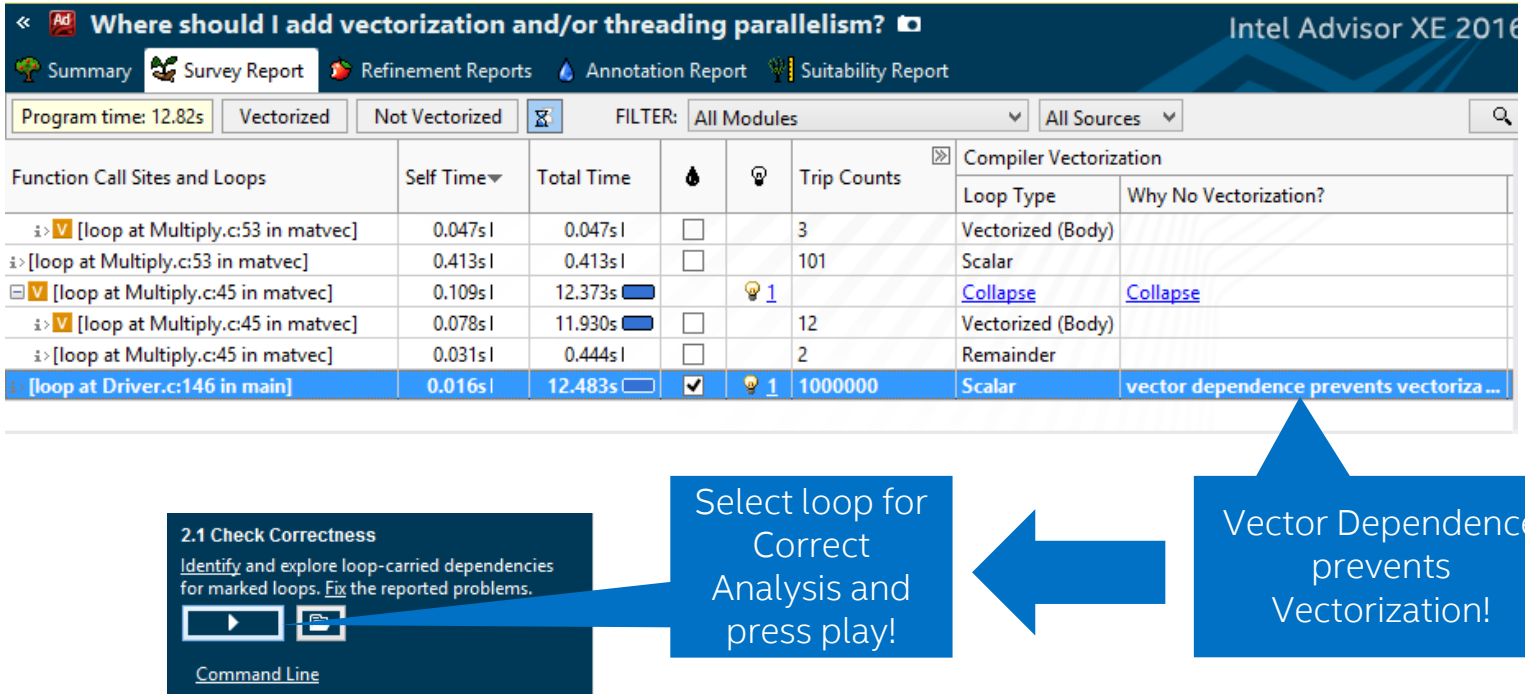
Verifying dependency with variables

- Is it possible to vectorize loop carried dependencies ?



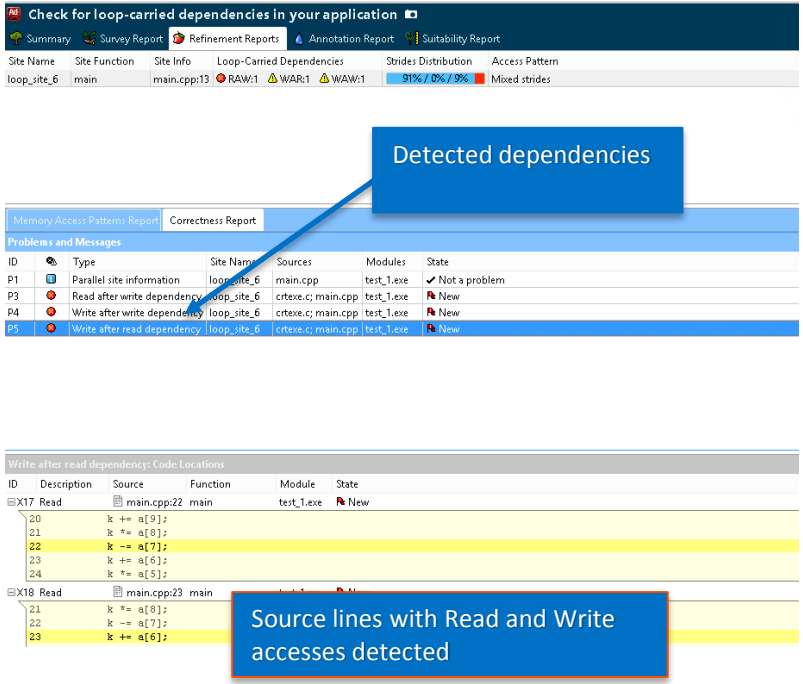
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

Is it safe to vectorize ?



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

HOW DOES INTEL® ADVISOR DISPLAY THE RESULTS ?



- Received recommendations to force vectorization of a loop:

1. Mark-up loop and check for REAL dependencies
2. Explore dependencies 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!

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

CACHE SIMULATION

New!

Utilize memory sub-system effectively

Intel® Advisor: Cache simulation in frames of MAP analysis (*)

- Memory Performance Insights
  - Explores efficiency of caching:
    - Cache Line Utilization
    - Misses
    - Evictions
  - Suggests next optimization steps

Read for ownership

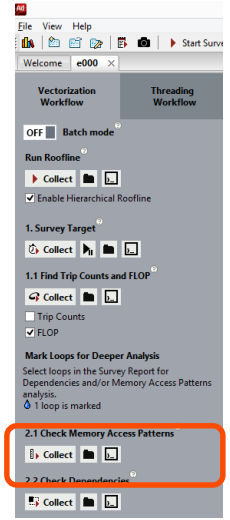
Site Name	Recommendations	Cache Line Utilization	Memory Loads	Memory Stores	Cache Misses	RFO Cache Misses	Dirty Evictions
loop_site_157	2 Potential excessive caching present	32 threads fit into cache	5922	94	433	47	0
loop_site_163		n/a	0	0	0	0	0
loop_site_170		32 threads fit into cache	54	2	14	1	0

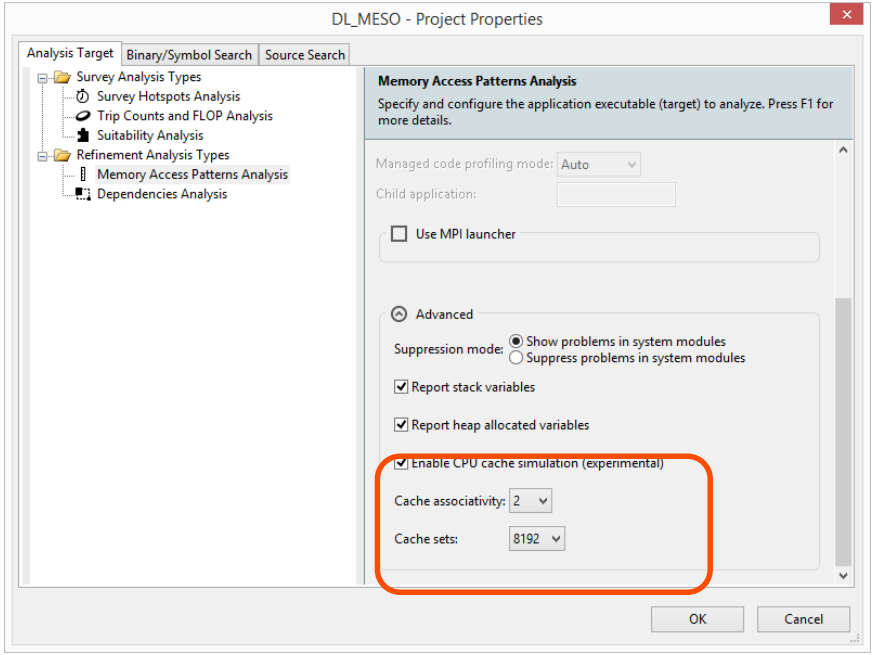
- * enable by additional option in Project Properties

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

MEMORY ACCESS (LATENCY FOCUSED) DEEP DIVE ANALYSIS

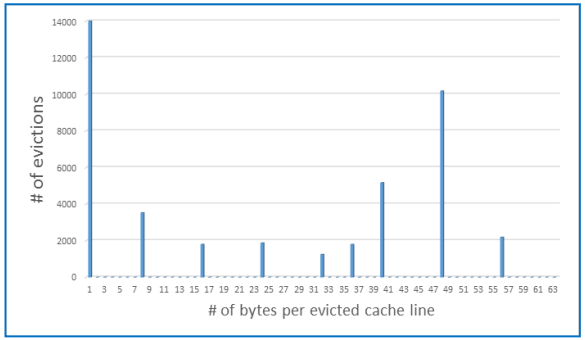
$ export ADVIXE_EXPERIMENTAL =cachesim

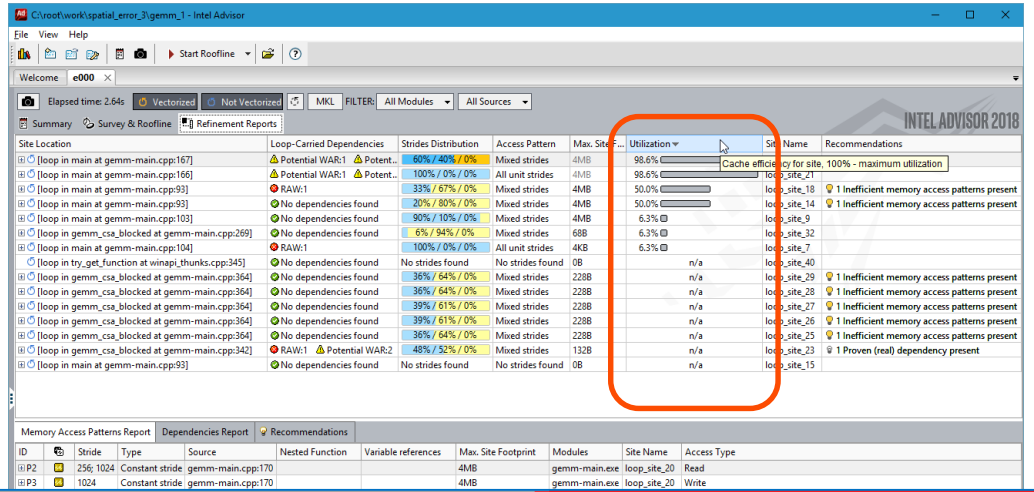




© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

LATENCY/SIMD (AOS/SOA) OPTIMIZATION & ANALYSIS WITH ADVISOR "MEMORY ACCESS PATTERN"





Site Location	Loop-Carried Dependencies	Strides Distribution	Access Pattern	Max. Site Footprint	Site Name	Recommendations	Cache Line ...	Memory Loads	Memory Stores	Cache Misses	RFO Cache Misses	Dirty Eviction
[loop in fCollisionBGKompparallel@368 at lbpBGK.cpp: ...	No information available	86% / 14% / 0%	Mixed strides	2MB	loop_site_7		Fit into cache...	3445340	640150	42893	10	22933

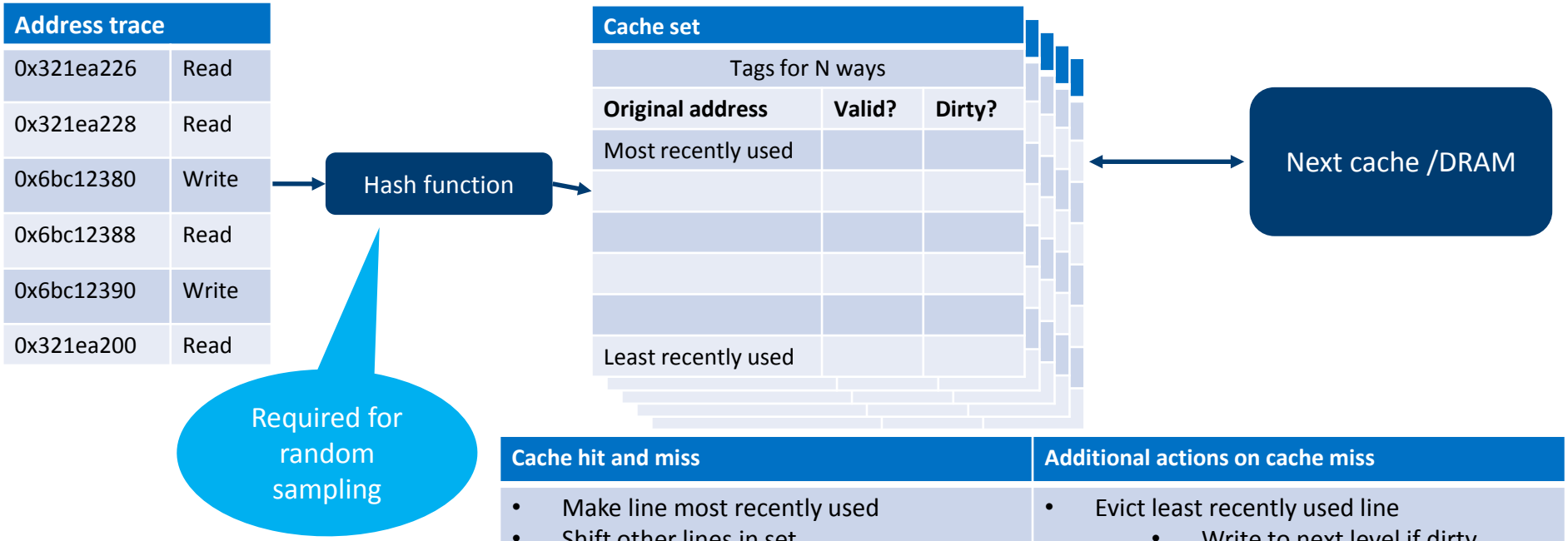
Memory Access Patterns Report | Dependencies Report | Recommendations

ID	Stride	Type	Source	Nested Function	Variable references	Max. Site Footprint	Modules
P1	25	Constant stride	lbpBGK.cpp:380		block 0xf4ce9d00d0 allocated at lbpSUB.cpp:78	15KB	libiomp5md.dll; slbe
P2	300	Constant stride	lbpBGK.cpp:382		block 0xf4e3d56040 allocated at lbpSUB.cpp:247	366KB	libiomp5md.dll; slbe

```
380             if(lbphi[il] != 11) {
381               for(ll=0; ll<3*lbsy.nf; ll++)
382                 interforce [ll] = lbinterforce[il*3*lbsy.nf+ll] + postequil*lbbdforce[ll];
```

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.
Intel Confidential

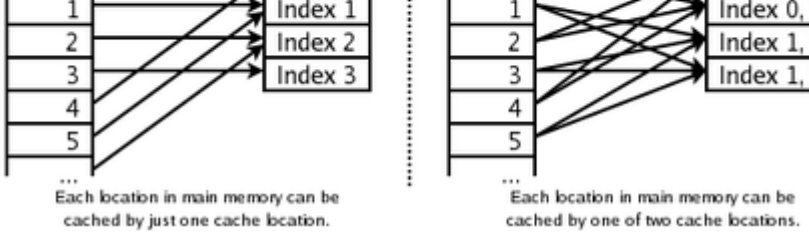
How cache simulator works?

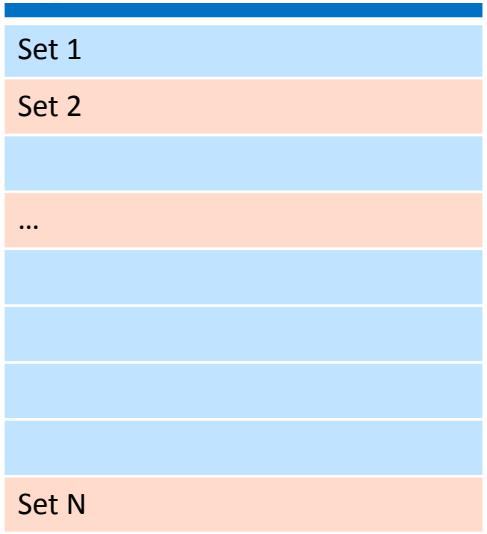


Cache hit and miss	Additional actions on cache miss
• Make line most recently used • Shift other lines in set • Mark line as valid • Mark line as dirty on write	• Evict least recently used line • Write to next level if dirty • Read line from next level

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

er?





Select some sets randomly and simulate

Ignore others!

Scale data to get final result

Total misses = Simulated misses * Set count / Sampled set count

Random sampling – cache sets are similar, no need to model the whole cache

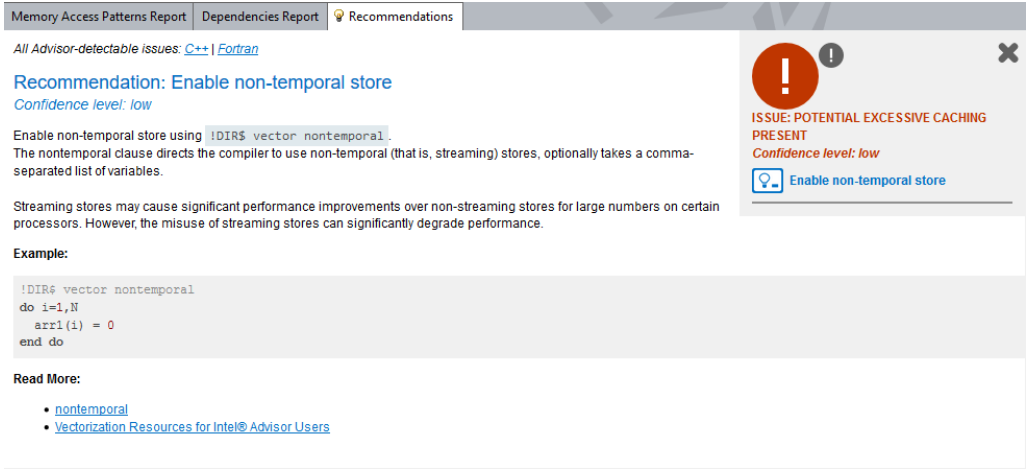
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

New!

Recommendation: enable NT-stores

Intel® Advisor: Cache simulation in frames of MAP analysis (*)

- Criteria
  - Write-only evictions from cache
  - No streaming stores generated
- Example - stream benchmark:
  - Kernels do not reuse the cache effectively
  - Intel Compiler (since 18.0) generate non-temporal stores for all of the kernels
- * enable by additional option in Project Properties



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

How to get cache informations out of your laptop ?

- This is possible to extract those information from your computer
  - cat /sys/devices/system/cpu/cpu[X]/cache/index[Y]/*
  - You can simulate the behavior of your cache
- You can also retrieve those information from specification
  - Simulate cache behavior for hardware not yet release
  - Optimize your cache blocking for next generation hardware

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

FEW COMMENTS

HOW CAN INTEL® ADVISOR HELP ?

What can prevent vectorization or reduce efficiency ?

1. Loop-carried dependencies

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

2. Pointer aliasing (compiler specific)

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

3. Loop structure, boundary condition

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

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

Intel® Advisor dependency analysis

4. Outer vs inner loops

Intel® Advisor MAP analysis

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

5. Indirect memory access or non unit-strides

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

6. Small trip counts not multiple of VL

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

Intel® Advisor Trip count

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

INTEL® ADVISOR 2018

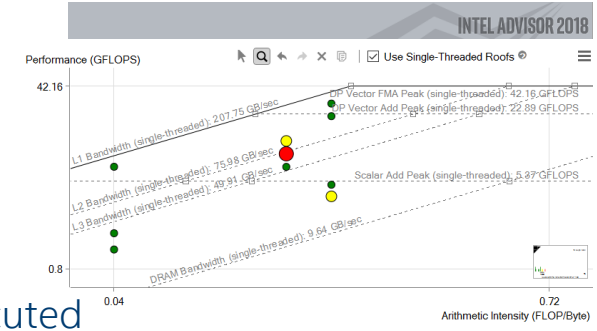
WHAT'S NEW?

New!

New! Roofline, Faster Analysis & More…

Intel® Advisor – Vectorization Optimization

- Roofline analysis helps you optimize effectively
  - Find high impact, but under optimized loops
  - Does it need cache or vectorization optimization?
  - Is a more numerically intensive algorithm a better choice?
- Faster data collection
  - Filter by module - Calculate only what is needed.
  - Track refinement analysis – Stop when every site has executed
- Make better decisions with more data, more recommendations
  - Intel MKL friendly – Is the code optimized? Is the best variant used?
  - Function call counts in addition to trip counts
  - Top 5 recommendations added to summary
  - Dynamic instruction mix – Expert feature shows exact count of each instruction
- Easier MPI launching
  - MPI support in the command line dialog



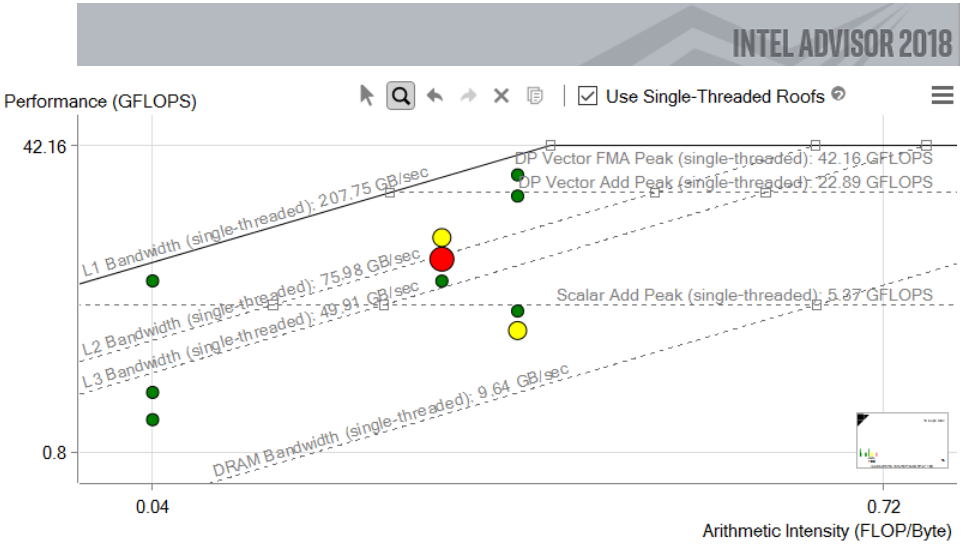
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

New!

Find Effective Optimization Strategies

Intel Advisor: Cache-aware roofline analysis

- Roofline Performance Insights
  - Highlights poor performing loops
  - Shows performance "headroom" for each loop
    - Which can be improved
    - Which are worth improving
  - Shows likely causes of bottlenecks
  - Suggests next optimization steps



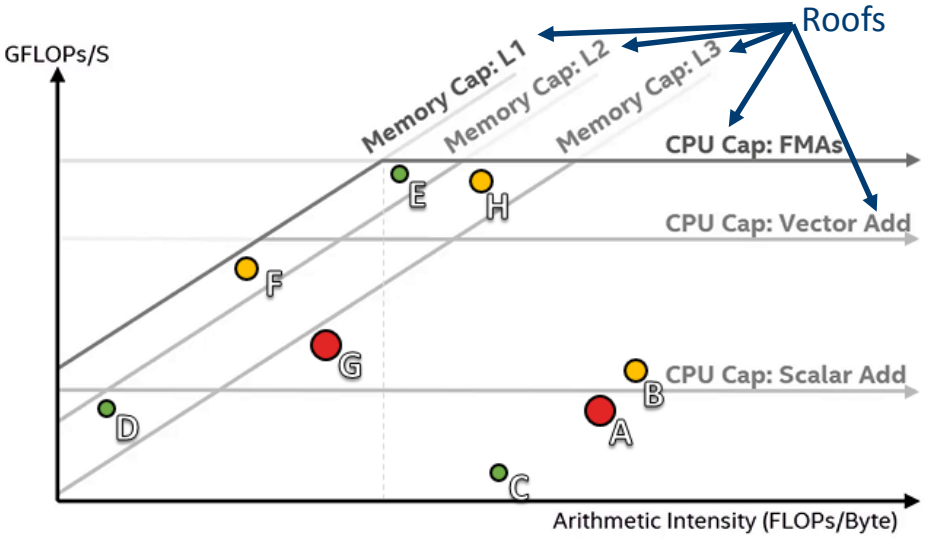
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

New!

Find Effective Optimization Strategies

Intel Advisor: Cache-aware roofline analysis

- Roofs Show Platform Limits
  - Memory, cache & compute limits
- Dots Are Loops
  - Bigger, red dots take more time so optimization has a bigger impact
  - Dots farther from a roof have more room for improvement
- Higher Dot = Higher GFLOPs/sec
  - Optimization moves dots up
  - Algorithmic changes move dots horizontally



Which loops should we optimize?

- A and G are the best candidates
- B has room to improve, but will have less impact
- E, C, D, and H are poor candidates

Roofline tutorial video

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

INTEL® ADVISOR 2017

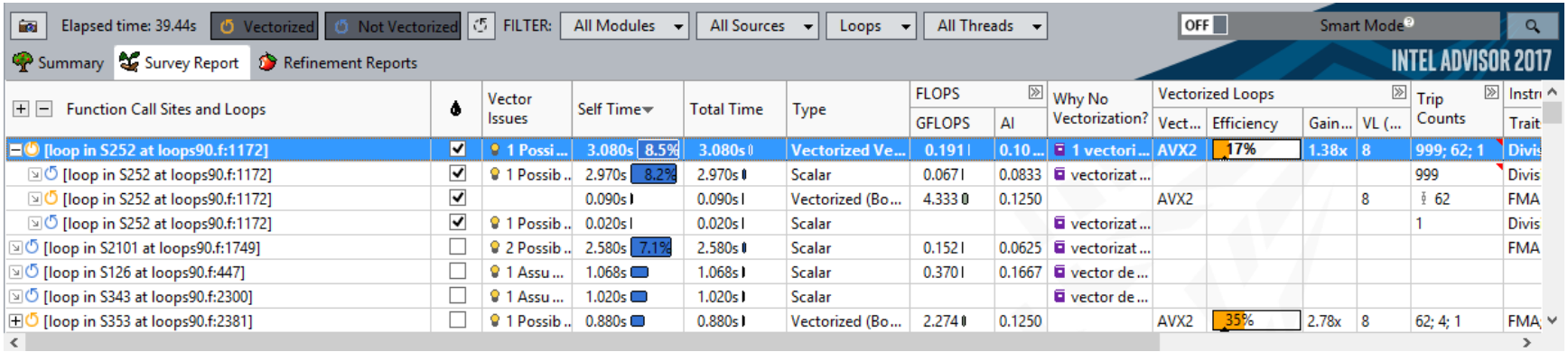
WHAT WAS NEW?

New!

NEW FOR 2017! AVX-512, FLOPS, & MORE…

Intel® Advisor – Vectorization Optimization

- Next Gen Intel® Xeon Phi™ Support
- Tune for AVX-512
  with or without AVX-512 hardware
- Precise FLOPS calculation
- Enhanced Memory Access Analysis
- Easier Selection of High Impact Loops
- Batch Mode Workflow Saves Time
- Fast Answers with Loop Analytics



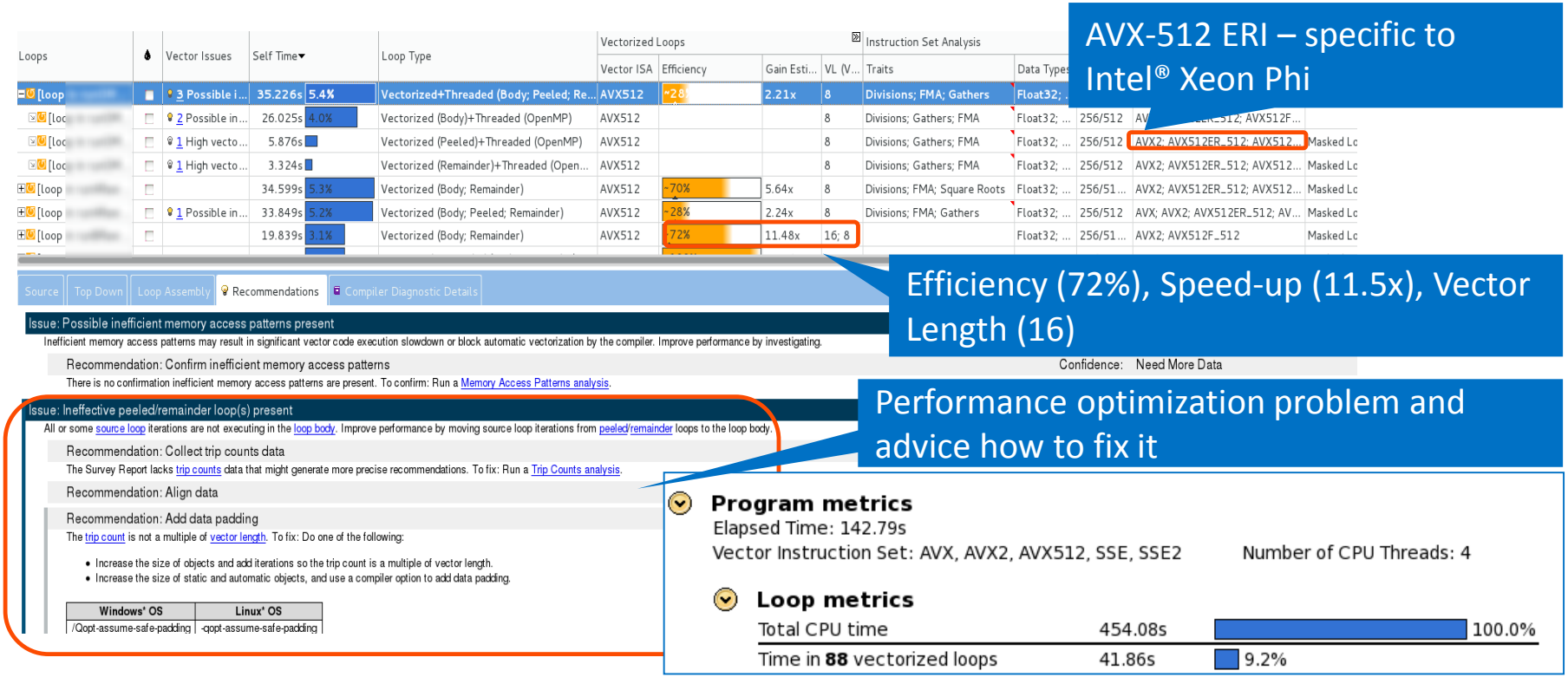
Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	FLOPS GFLOPS	FLOPS AI	Why No Vectorization?	Vectorized Loops Vect…	Vectorized Loops Efficiency	Vectorized Loops Gain…	Vectorized Loops VL (…	Trip Counts	Instr… Trait
[loop in S252 at loops90.f:1172]	1 Possi…	3.080s 8.5%	3.080s	Vectorized Ve…	0.191	0.10…	1 vectori…	AVX2	17%	1.38x	8	999; 62; 1	Divis
[loop in S252 at loops90.f:1172]	1 Possib…	2.970s 8.2%	2.970s	Scalar	0.067	0.0833	vectorizat…					999	Divis
[loop in S252 at loops90.f:1172]		0.090s	0.090s	Vectorized (Bo…	4.333	0.1250		AVX2			8	62	FMA
[loop in S252 at loops90.f:1172]	1 Possib…	0.020s	0.020s	Scalar			vectorizat…					1	Divis
[loop in S2101 at loops90.f:1749]	2 Possib…	2.580s 7.1%	2.580s	Scalar	0.152	0.0625	vectorizat…						FMA
[loop in S126 at loops90.f:447]	1 Assu…	1.068s	1.068s	Scalar	0.370	0.1667	vector de…						
[loop in S343 at loops90.f:2300]	1 Assu…	1.020s	1.020s	Scalar			vector de…						
[loop in S353 at loops90.f:2381]	1 Possib…	0.880s	0.880s	Vectorized (Bo…	2.274	0.1250		AVX2	35%	2.78x	8	62; 4; 1	FMA

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

NEXT GEN INTEL® XEON PHI™ PROCESSOR SUPPORT

New!

Vectorization Advisor runs on and optimizes for Intel® Xeon Phi Processors



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

START TUNING FOR AVX-512 WITHOUT AVX-512 HARDWARE

Intel® Advisor - Vectorization Advisor

New!

- 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)

© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

PRECISE REPEATABLE FLOPS METRICS

Intel® Advisor – Vectorization Optimization

New!

- FLOPS by loop and function
- All recent Intel processors (not co-processors)
- Instrumentation (count FLOPs) plus sampling (time with low overhead)
- Adjusted for masking with AVX-512 processors

INTEL ADVISOR 2017

Function Call Sites and Loops	FLOPS						
	GFLOPS	AI	L1 GB/s	GFLOP	FLOP Per Iteration	L1 GB	L1 Bytes Per Iteration
[loop in matvec at Multiply.c:69]	0.826	0.1633	5.0586	3.0720	32	18.8160	196
[loop in matvec at Multiply.c:60]	0.912	0.1633	5.5853	3.0720	32	18.8160	196
[loop in matvec at Multiply.c:69]	1.248	0.2500	4.9920	1.3440	4	5.3760	16
[loop in matvec at Multiply.c:60]	1.592	0.2500	6.3699	1.3440	4	5.3760	16
[loop in matvec at Multiply.c:69]	3.055	0.2500	12.2205	0.0960	16	0.3840	64
[loop in matvec at Multiply.c:60]	6.282	0.2500	25.1279	0.0960	16	0.3840	64

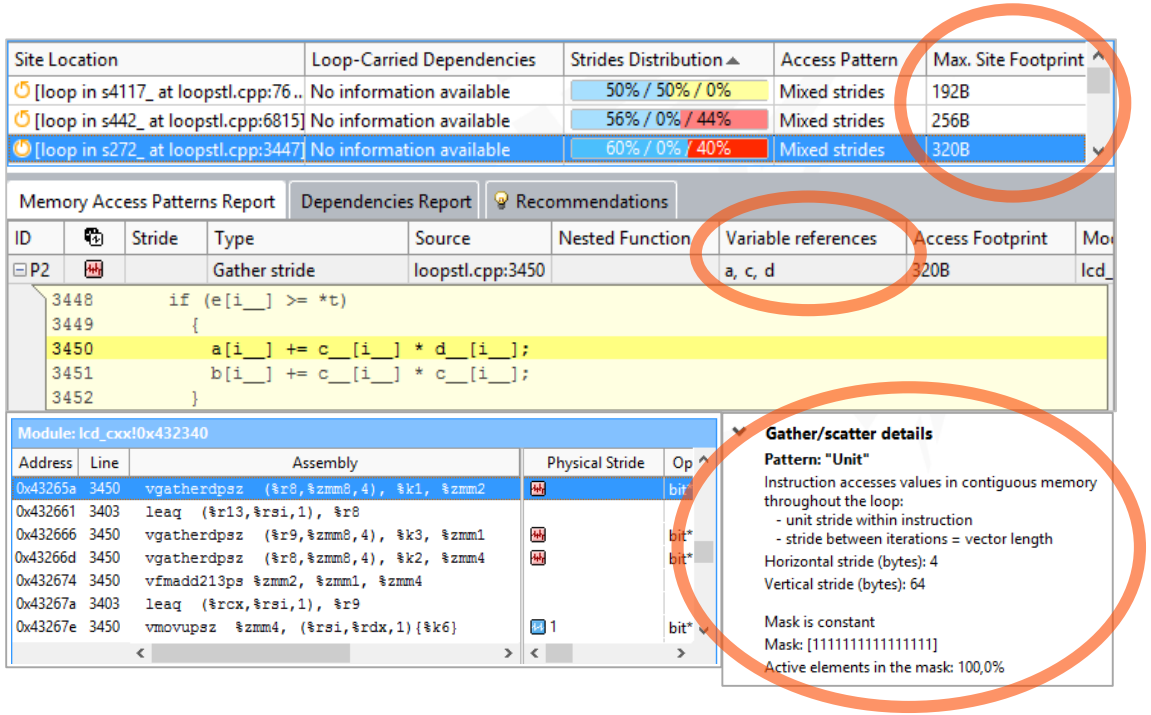
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

ENHANCED MEMORY ACCESS ANALYSIS

New!

Are you bandwidth or compute limited?

- Measure Footprint
  - Compare to cache size
    Does it fit in L2 cache?
- Variable References
  - Map data to variable names for easier analysis
- Gather/Scatter
  - Detect unneeded gather/scatters that reduce performance



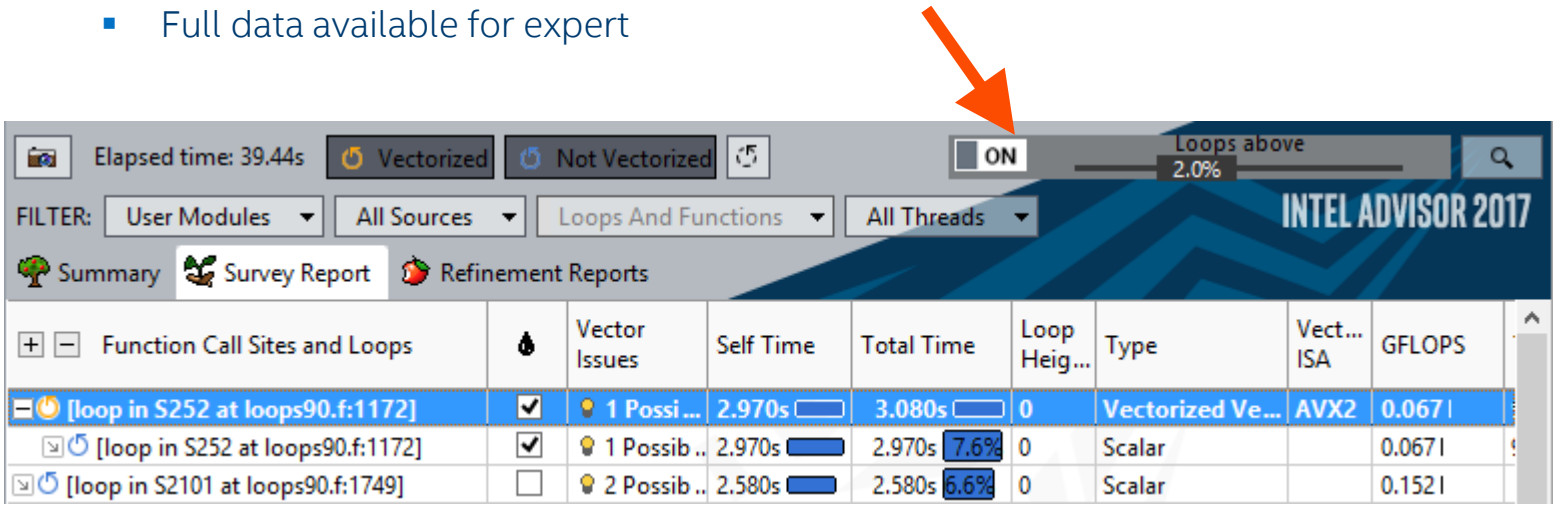
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

EASIER SELECTION OF HIGH IMPACT LOOPS

New!

Smart mode helps you prioritize quickly

- Smart Mode On
  - Simplified column layout – heuristics prioritize high impact loops
  - Slider adjusts filter threshold to hide low impact loops
- Smart Mode Off
  - Full data available for expert

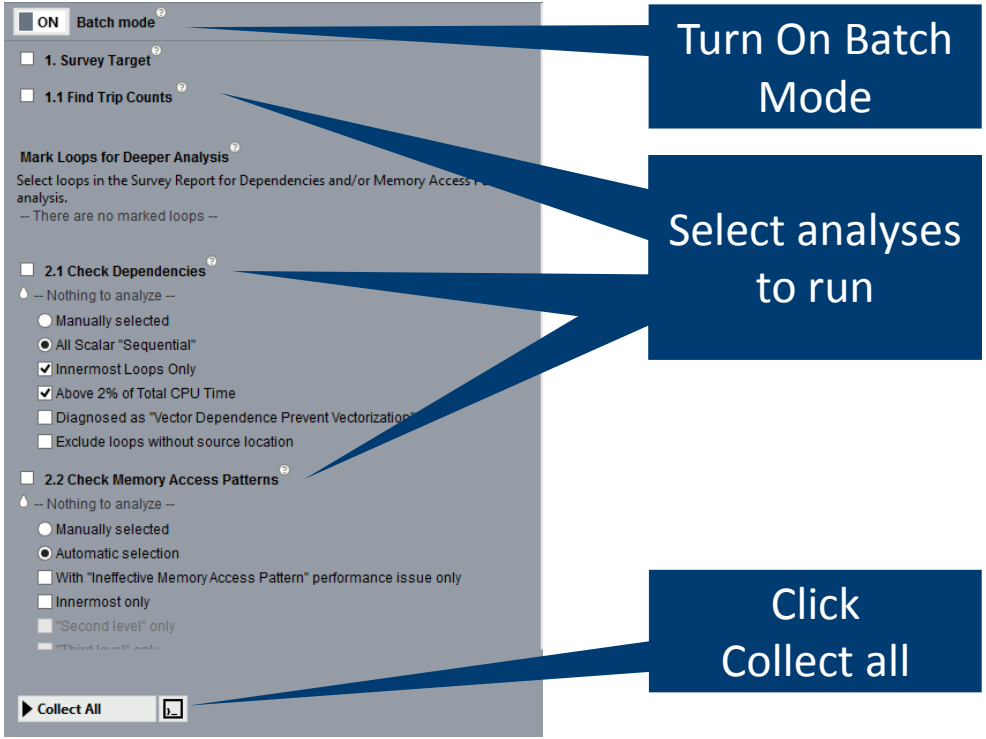


© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

BATCH MODE WORKFLOW SAVES TIME

Intel® Advisor - Vectorization Advisor

New!



- Start several analyses with a single click
- Contains pre-selected criteria for advanced analyses

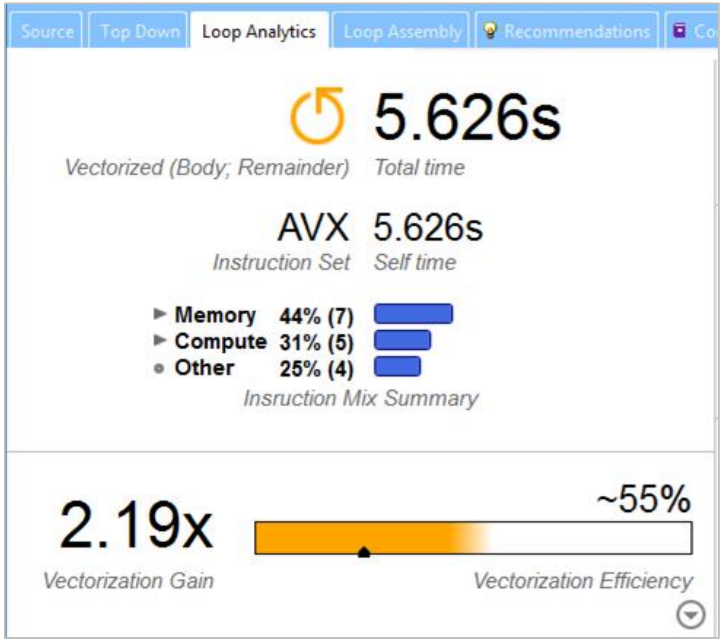
© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.

LOOP ANALYTICS

Get the data you need to plan optimization

New!

- Fast Answers
  - How much performance can be gained?
  - Is it worth optimizing?
  - Static analysis estimate:
    Is it memory bound or compute bound?



© 2017 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.
For more complete information about compiler optimizations, see our Optimization Notice.


intel
Software