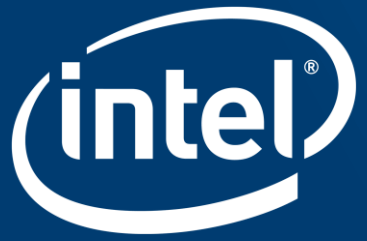


INTEL® ADVISOR FOR OPENMP ON CPU AND INTEGRATED GPU

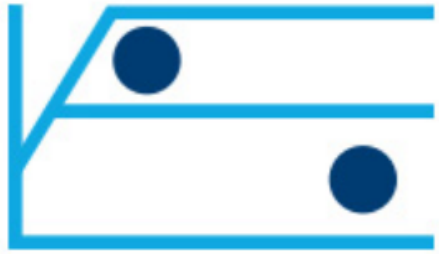
Kirill Rogozhin, November 2019

Intel Advisor

Roofline: Characterize and optimize on **CPU**

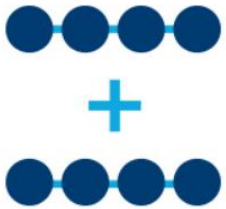
Offload Advisor: identify best offload candidates

Roofline: Characterize and optimize on **GPU**



```
omp target
omp teams
omp map...
```

Vectorization



```
omp simd
omp declare simd
...
```

Memory

Threading

```
omp parallel_for
omp task
...
```

Advisor Roofline

Characterize and optimize CPU code

What is the Roofline Model?

Characterization of your application performance in the context of the hardware

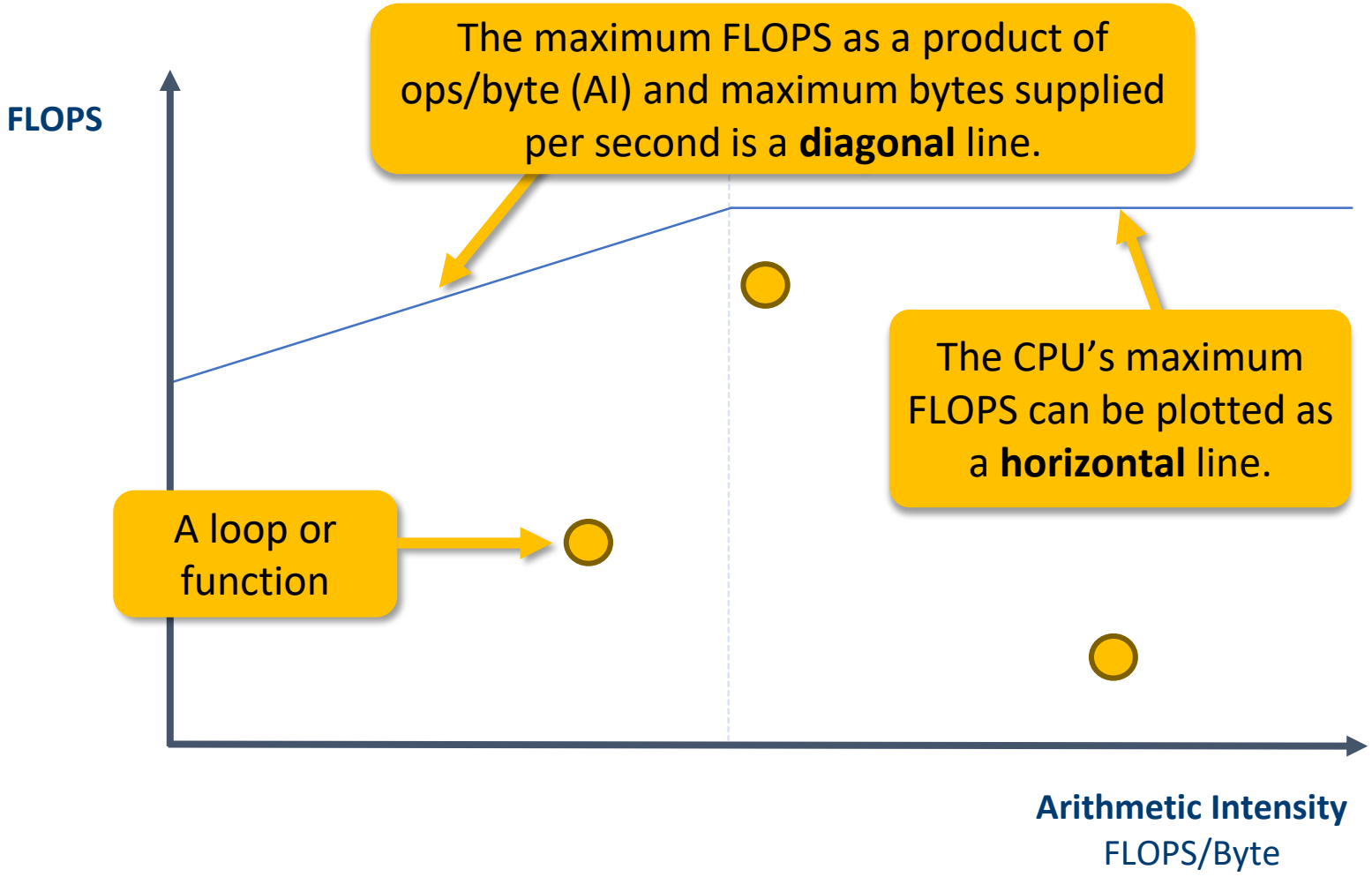
It uses two simple metrics

- Flop count
- Bytes transferred

2 Operations

$$a_i = b_i + c_i * d_i$$

1W+3R = 4*4bytes = 16 bytes



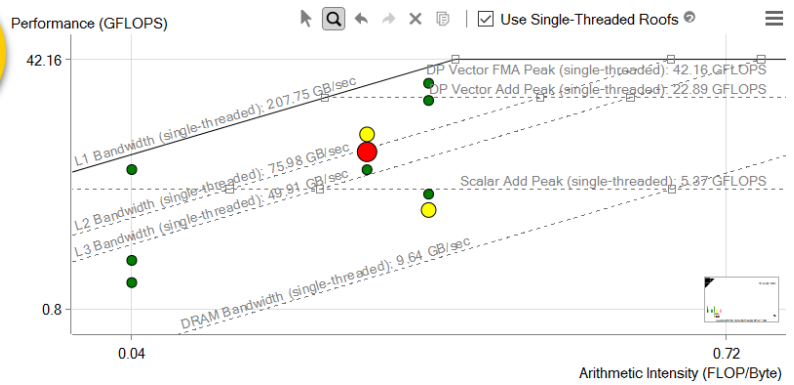
The Roofline Chart in Intel® Advisor

1

A single button or CLI command runs the Survey and FLOPS analyses to generate the Roofline chart.

2

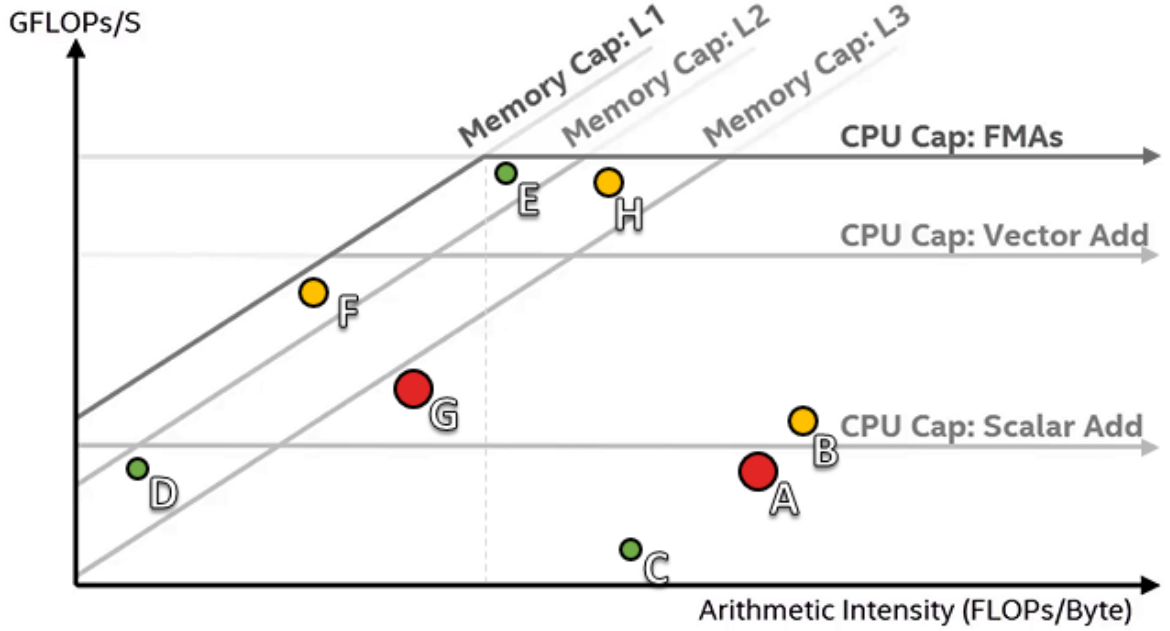
3



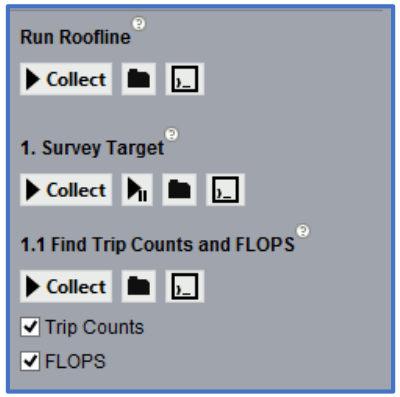
Focus on big dots with headroom.

- Intel® Advisor sizes and color-codes dots by relative time taken.
- Space between dots and their uppermost limits is room to improve.

Roofs above a dot indicate potential sources of bottlenecks.



Getting Roofline data in Intel® Advisor



Roofline : Axis X: **AI** = #**FLOP** / #**Bytes** Axis Y: **FLOP/S** = #**FLOP** (mask aware) / #**Seconds**	Overhead
Step 1: Survey (-collect survey) - Provide **#Seconds** - *Root access not needed* - User mode sampling, non-intrusive.	**1x**
Step 2: FLOPS *(-collect tripcounts –flops)* - Provide #**FLOP**, #**Bytes**, AVX-512 Mask - *Root access not needed* - Precise, instrumentation based, count number of instructions	**5-10x**

Questions to answer with Roofline

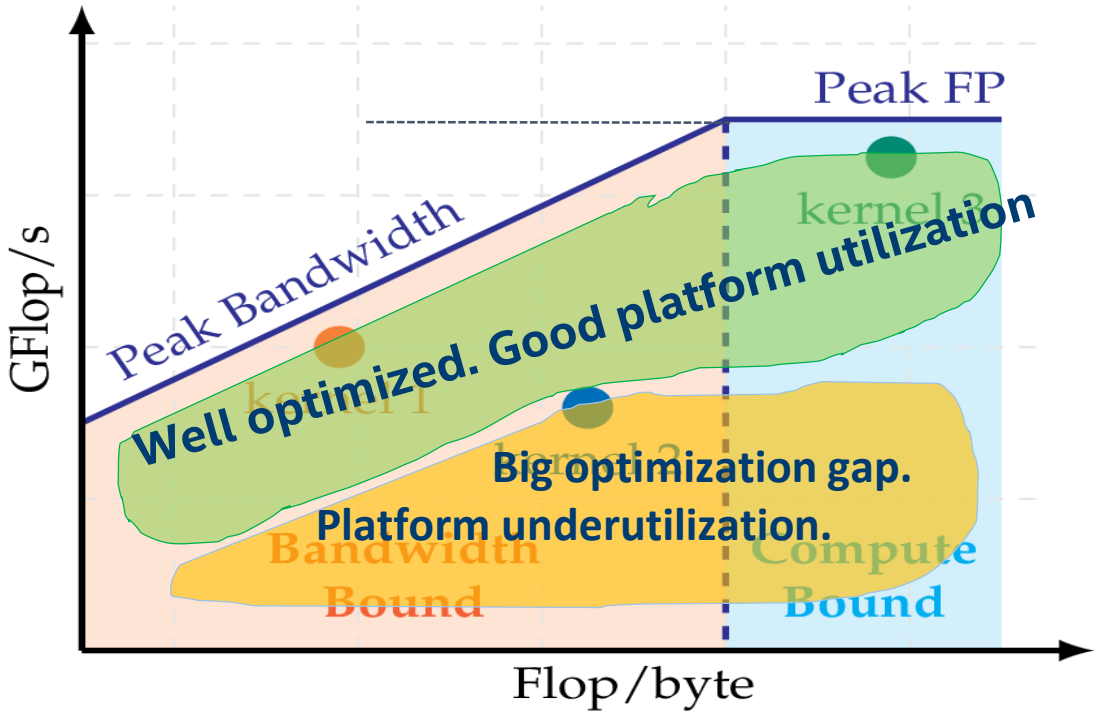
1 **Am I doing well? How far am I from the pick?**

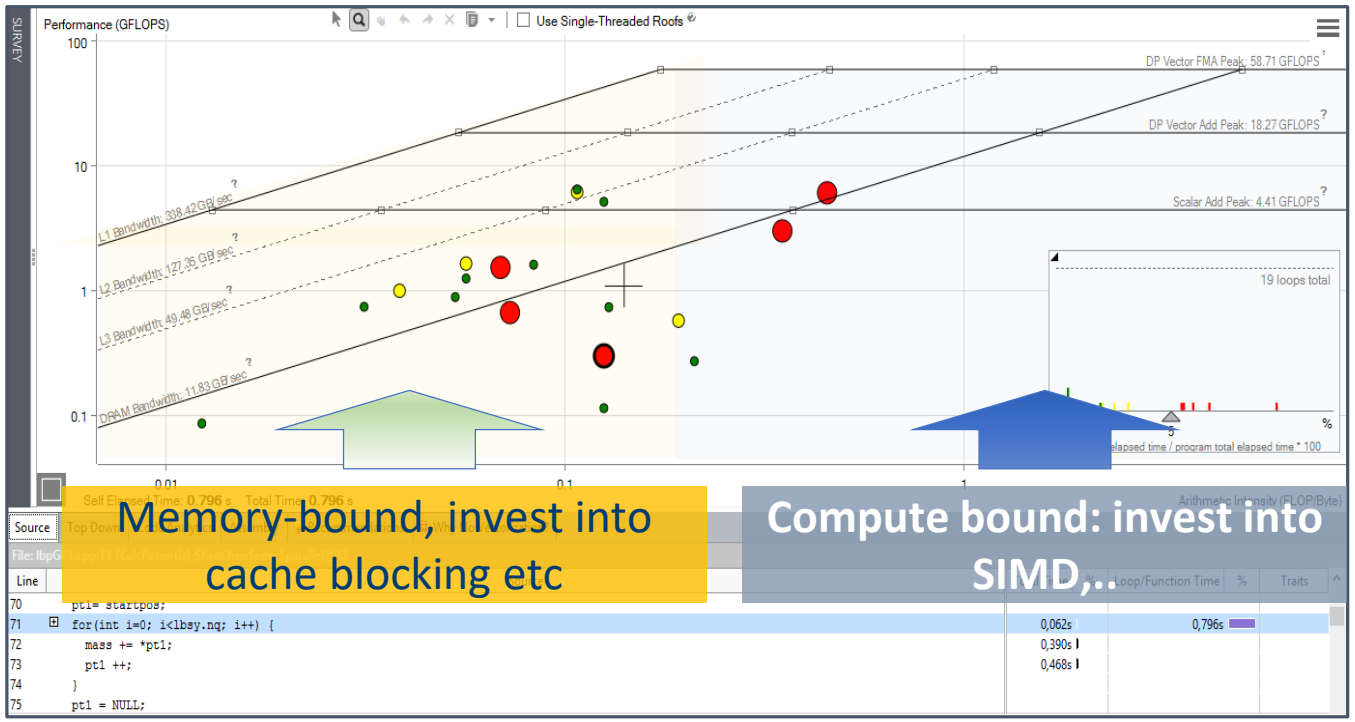
(do I utilize hardware well or not?)

2 **Final** Bottleneck?

(where will be my limit after I done all optimizations?)

Long-term ROI, optimization strategy



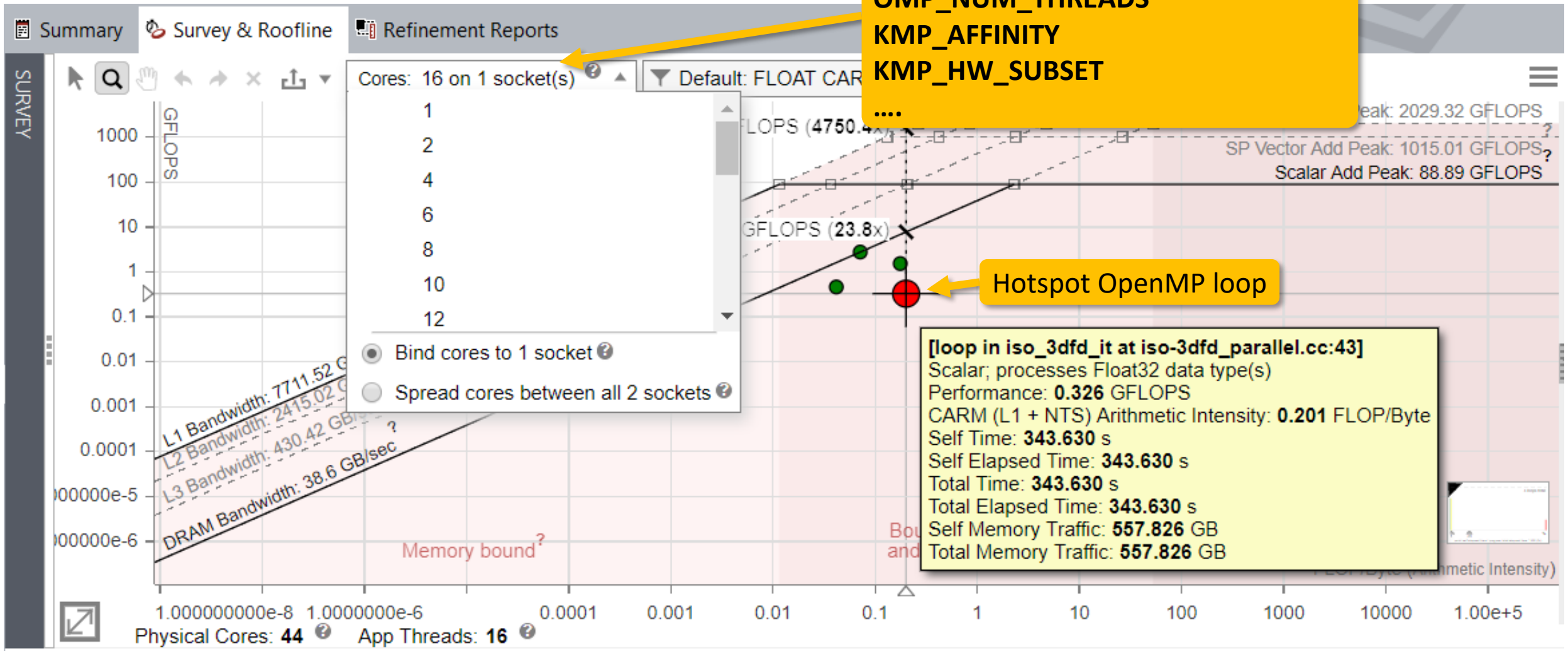


Advisor Roofline

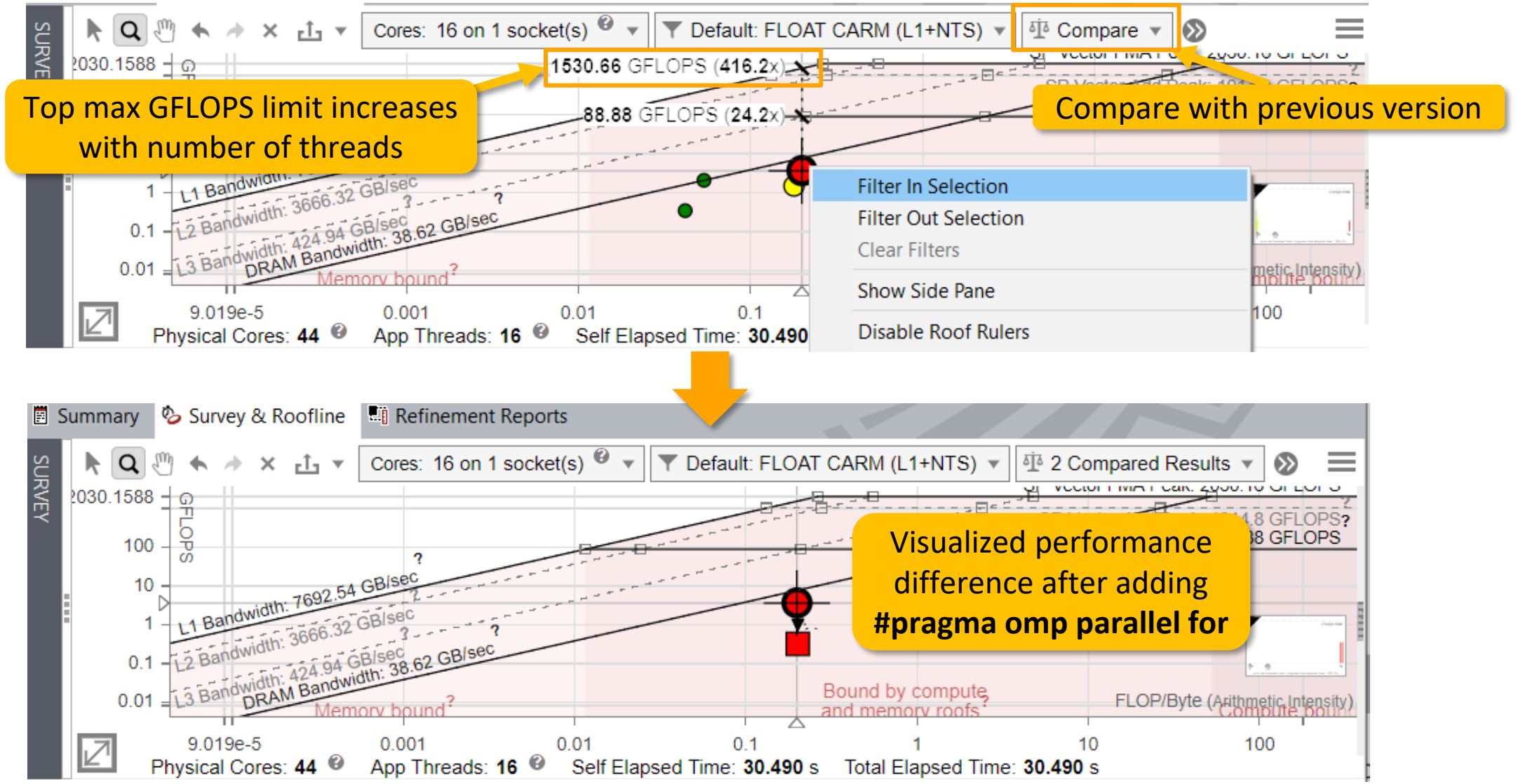
Characterize and optimize CPU code: Threading

Roofline: iso3dfd example

Adjust roofs to #cores, ranks, NUMA in your OpenMP application:
OMP_NUM_THREADS
KMP_AFFINITY
KMP_HW_SUBSET
….

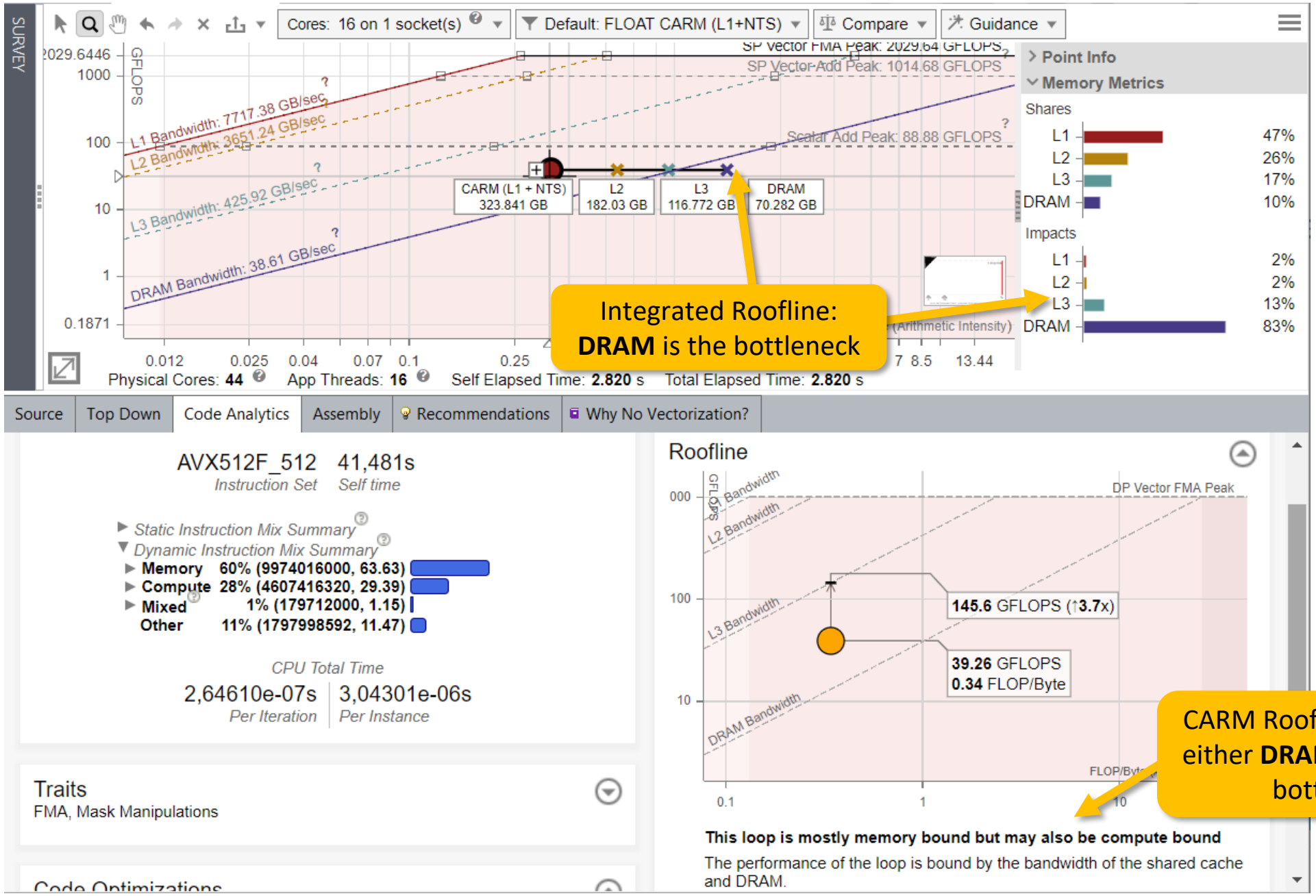


Roofline: iso3dfd example



Advisor Roofline

Characterize and optimize CPU code: Memory


SURVEY
Cores: 16 on 1 socket(s)
Default: FLOAT CARM (L1+NTS)
Compare
Guidance
GFLOPS
2029.6446
1000
100
10
1
0.1871
SP Vector FMA Peak: 2029.64 GFLOPS
SP Vector Add Peak: 1014.68 GFLOPS
Scalar Add Peak: 88.88 GFLOPS
L1 Bandwidth: 7717.38 GB/sec
L2 Bandwidth: 3651.24 GB/sec
L3 Bandwidth: 425.92 GB/sec
DRAM Bandwidth: 38.61 GB/sec
CARM (L1 + NTS)
323.841 GB
L2
182.03 GB
L3
116.772 GB
DRAM
70.282 GB
0.012
0.025
0.04
0.07
0.1
0.25
7
8.5
13.44
(Arithmetic Intensity)
Physical Cores: 44
App Threads: 16
Self Elapsed Time: 2.820 s
Total Elapsed Time: 2.820 s
Point Info
Memory Metrics
Shares
L1 47%
L2 26%
L3 17%
DRAM 10%
Impacts
L1 2%
L2 2%
L3 13%
DRAM 83%
Integrated Roofline:
DRAM is the bottleneck
Source
Top Down
Code Analytics
Assembly
Recommendations
Why No Vectorization?
AVX512F_512
Instruction Set
41,481s
Self time
Static Instruction Mix Summary
Dynamic Instruction Mix Summary
Memory 60% (9974016000, 63.63)
Compute 28% (4607416320, 29.39)
Mixed 1% (179712000, 1.15)
Other 11% (1797998592, 11.47)
CPU Total Time
2,64610e-07s
Per Iteration
3,04301e-06s
Per Instance
Traits
FMA, Mask Manipulations
Code Optimizations
Roofline
GFLOPS
000
100
10
SP Bandwidth
L2 Bandwidth
L3 Bandwidth
DRAM Bandwidth
DP Vector FMA Peak
145.6 GFLOPS (↑3.7x)
39.26 GFLOPS
0.34 FLOP/Byte
0.1
1
10
This loop is mostly memory bound but may also be compute bound
The performance of the loop is bound by the bandwidth of the shared cache and DRAM.
CARM Roofline Guidance:
either DRAM or LLC is the bottleneck

Memory access pattern analysis

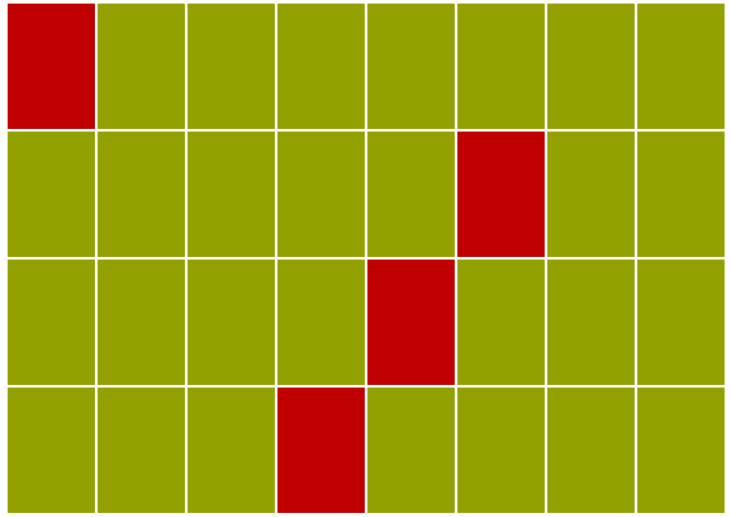
How should I access data ?

Unit stride access are faster

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

Constant stride are usually worse

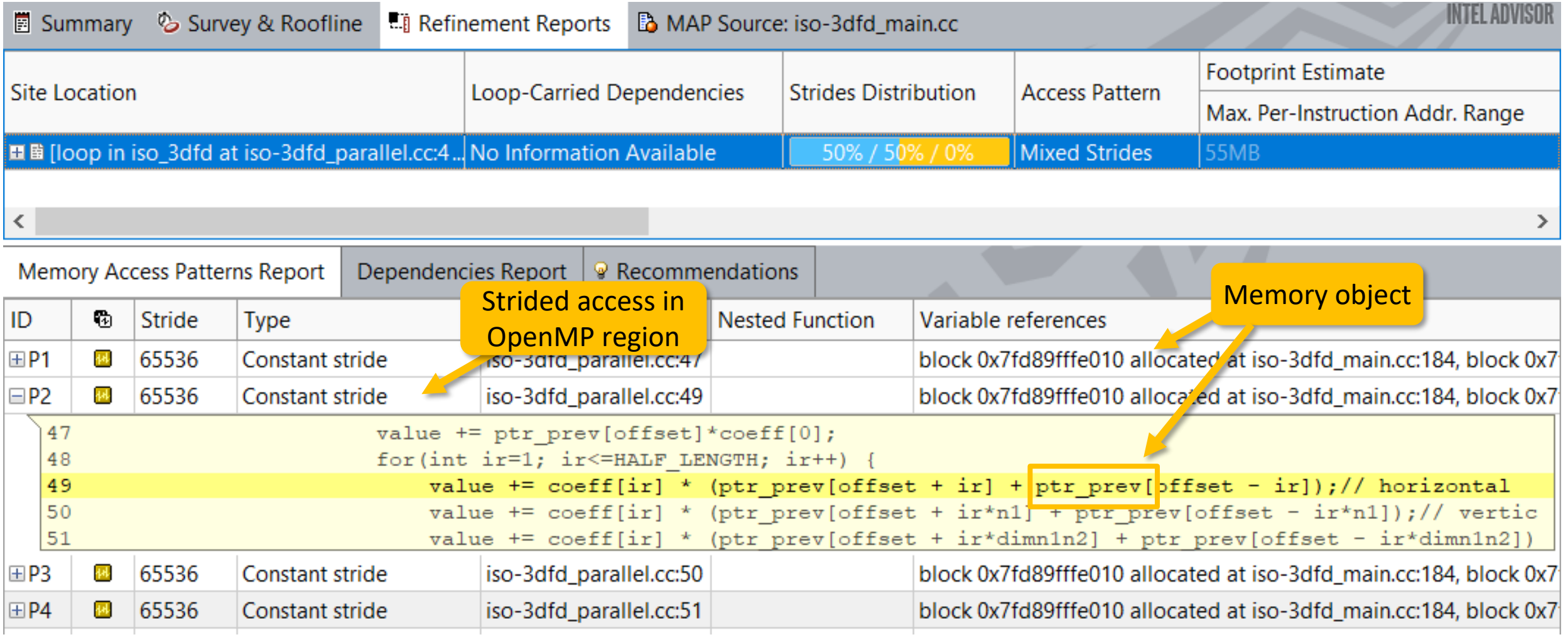
```
for (i=0; i<N; i+=2)
    A[i] = B[i]*d
```



Non predictable access are usually bad

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

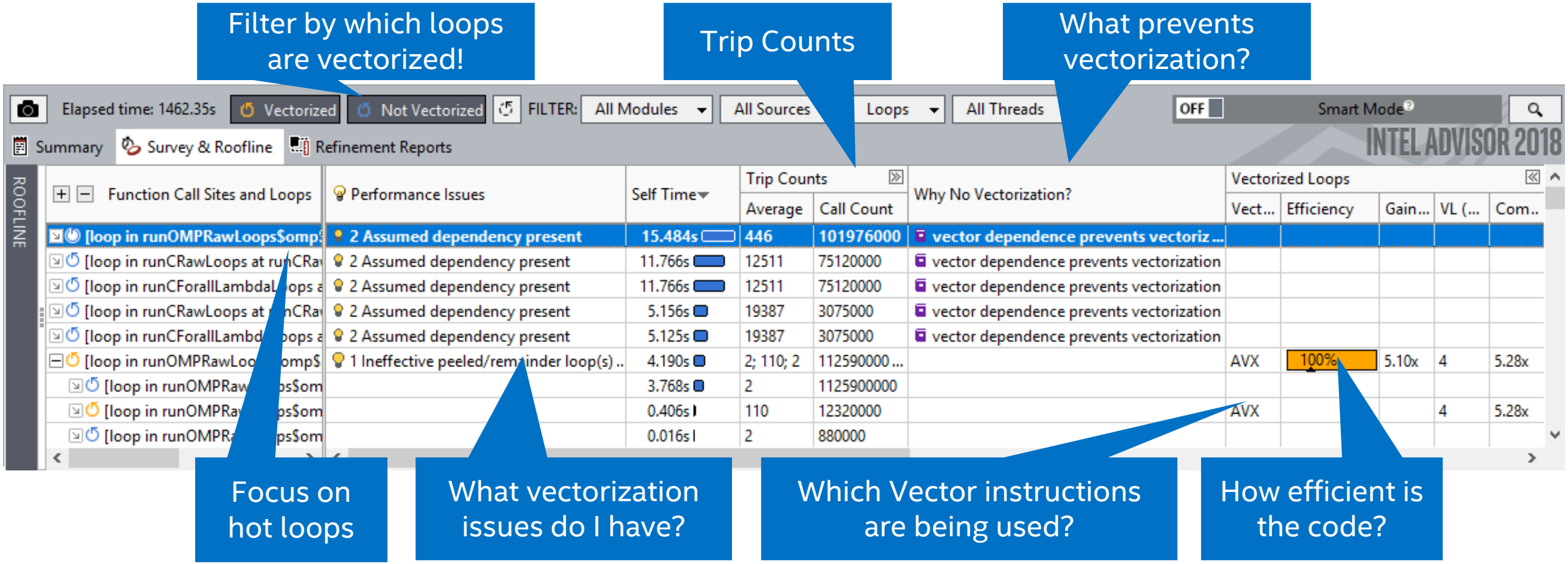
Memory Access Patterns



Advisor Roofline

Characterize and optimize CPU code: Vectorization

Advisor Vectorization analysis



Enabling Vectorization

1

Vector Issues	Self Time	Total Time	Type	W
2 Assumed dependency present	20.030s	20.030s	Scalar Versions	
	13.508s	13.508s	Scalar	
	6.895s	27.750s	Scalar	

2

Check dependencies

Refinement Reports

	Loop-Carried Dependencies
h_4_v253.cpp:183]	No dependencies found
_3_1_5_kernel_max.cpp:80]	No dependencies found

3

Use **#pragma omp simd**

Vector Issues	Self Time	Total Time	Type	Vectorized Loops			
				Vector ISA	Efficiency	Gain ...	VL (V...
	10.507s	22.989s	Scalar				
2 Possible inef...	1.762s	3.190s	Vectorized Ver...	AVX512	73%	5.84x	8

Offload Advisor

Define regions to be offloaded

Offload Advisor Summary

Intel® Advisor Beta, build 604296

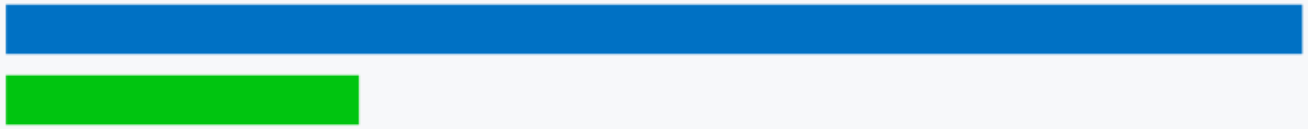
Intel® Advisor Beta
OFFLOAD ADVISOR

Summary | Offloaded Regions | Non Offloaded Regions | Call Tree | Configuration | Logs

Speed Up for Accelerated Code	4.2x	Number of Offloads	2	Fraction of Accelerated Code	95%

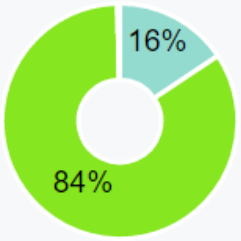
Program metrics

Original	7.04s
Accelerated	1.92s

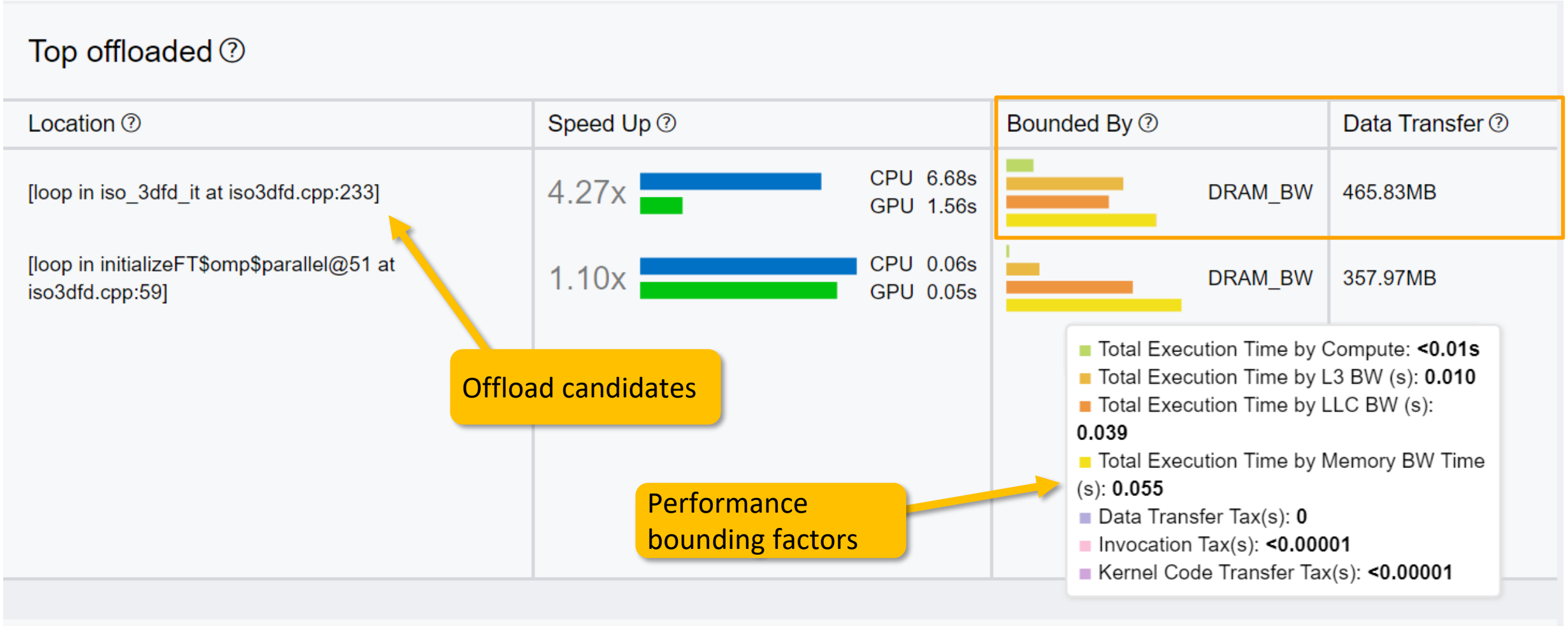


Target Platform	**Gen11 GT2**
Number of Offloads	2
Speed Up for Accelerated Code	4.2x
Amdahl's Law Speed Up	3.6x
Fraction of Accelerated Code	95%

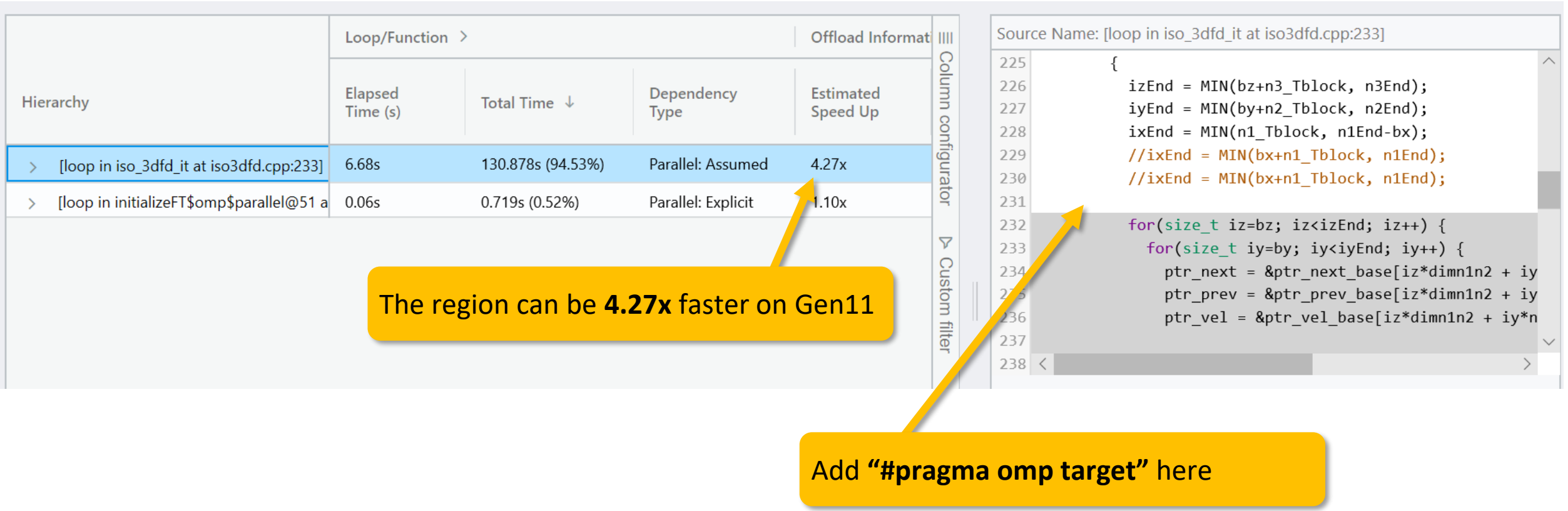
Time on Host	0.30s
Time on Accelerator	1.61s
Data Transfer Tax	0s
Invocation Tax	<0.01s
Code Transfer Tax	<0.01s



Explore offload candidates



Offloaded regions details and source view



Data transfer and memory mapping

`map(to:a)` `map(from:a)` `map(tofrom:a)`

Data Transfer

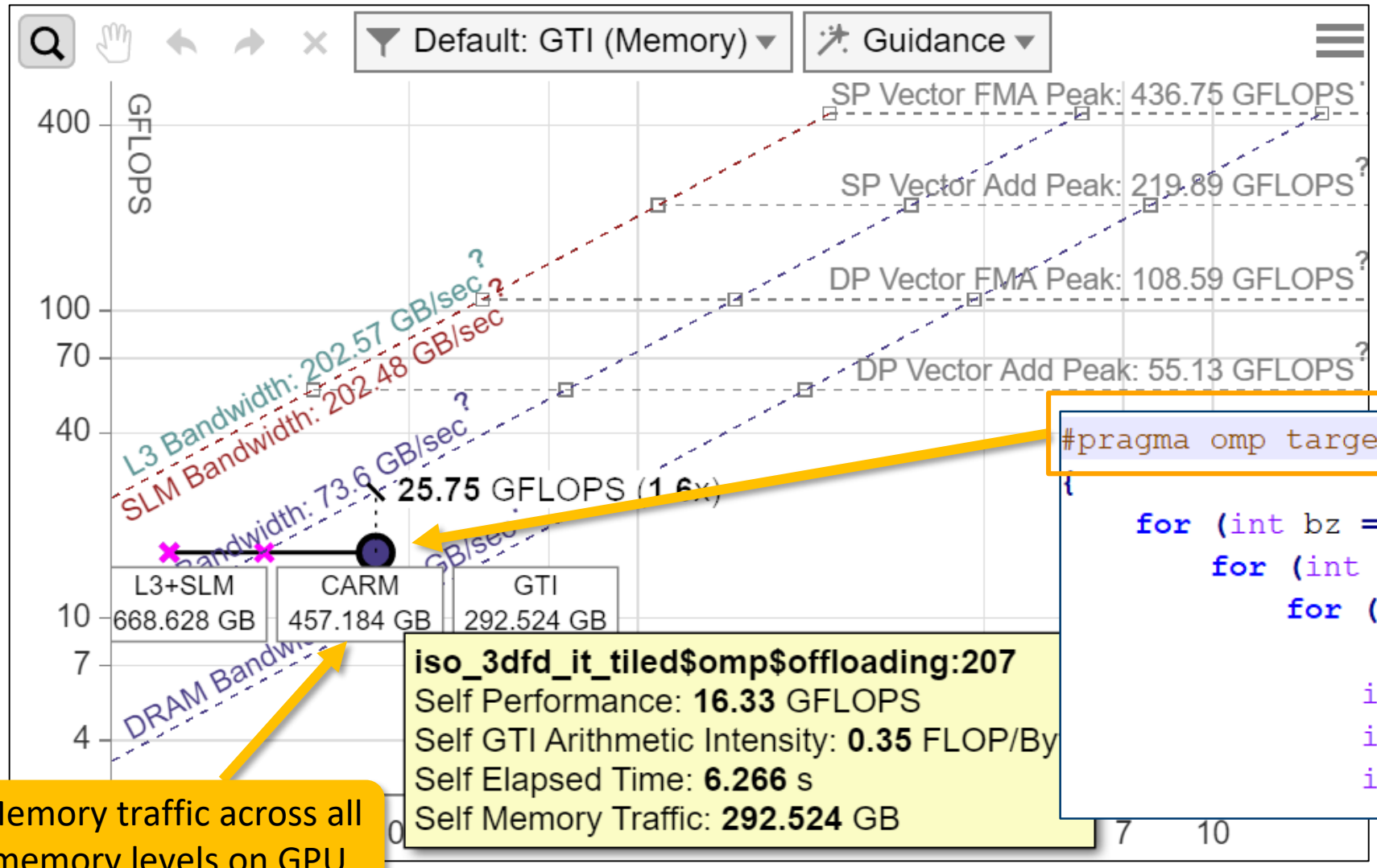
Total Da Transfer (MB)	Total Data Transferred from CPU to GPU (MB)	Total Data Transferred from GPU to CPU (MB)	Memory Mapped To Device (MB)	Memory Mapped From Device (MB)	Memory Mapped ToFrom Device (MB)
465.83MB	232.91	232.91	0	0	232.91494
357.97MB	116.47	241.50	0	125.02630	116.46976

Total estimated data traffic

GPU Roofline

Characterize and optimize GPU code

Roofline for GPU



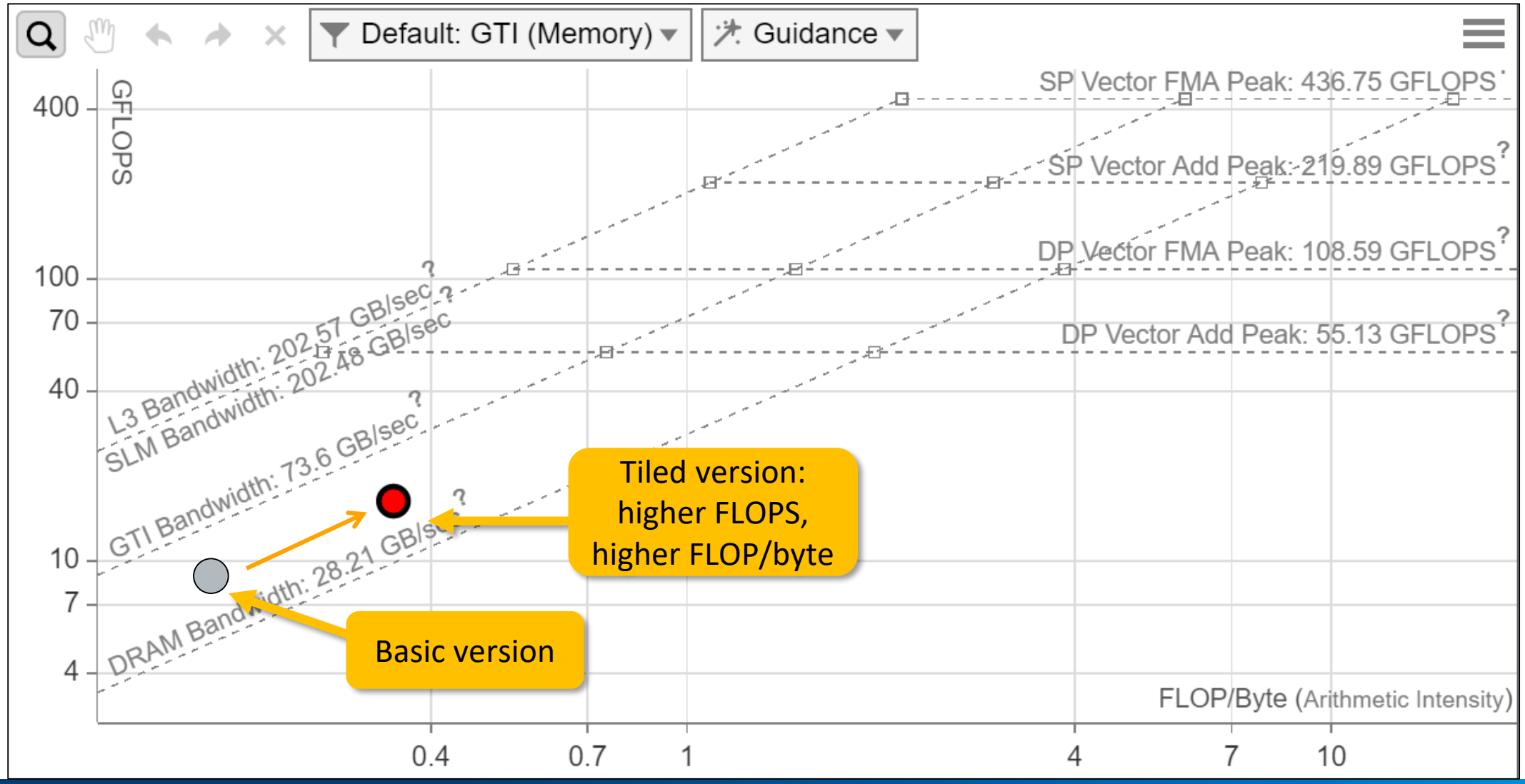

```
#pragma omp target teams distribute collapse(3)
{
    for (int bz = HALF_LENGTH; bz<n3End; bz += n
        for (int by = HALF_LENGTH; by<n2End; by
            for (int bx = HALF_LENGTH; bx<n1End;

                int izEnd = MIN(bz + n3_Tblock,
                int iyEnd = MIN(by + n2_Tblock,
                int ixEnd = MIN(bx + n1_Tblock,
```

Memory traffic across all memory levels on GPU

- Preview feature, supports Gen9 GPU
- Command line data collection
- HTML export as UI

GPU Roofline

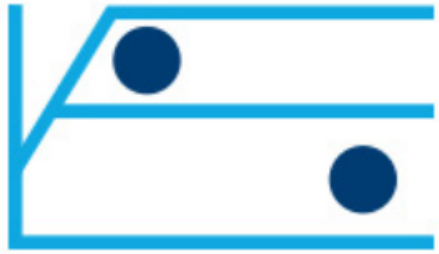


Intel Advisor

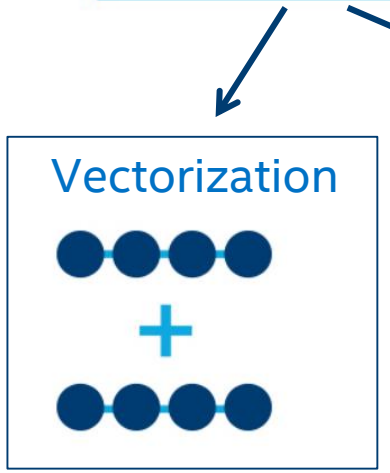
Roofline: Characterize and optimize on **CPU**

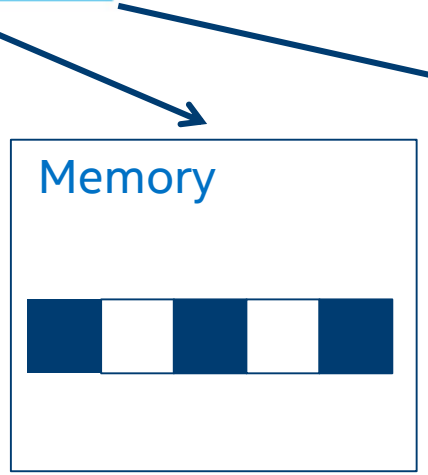
Offload Advisor: identify best offload candidates

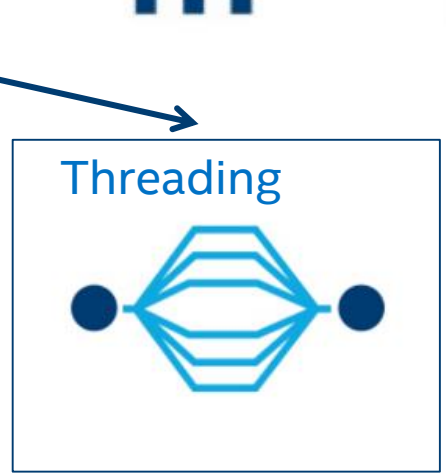
Roofline: Characterize and optimize on **GPU**



```
#pragma omp target
omp teams
...
```

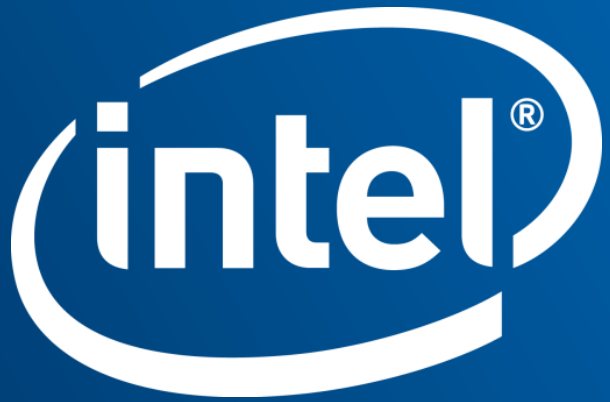







```
#pragma omp simd
#pragma omp declare simd
...
```

```
#pragma omp simd
#pragma omp declare simd
...
```


intel