

Optimization of Compute-Bound Applications

Stanford University, ME344 — Summer 2018

Andrey Vladimirov, Colfax International

Compute-Bound Performance

Theoretical Peak Performance of an Intel Xeon Gold 6128 CPU =

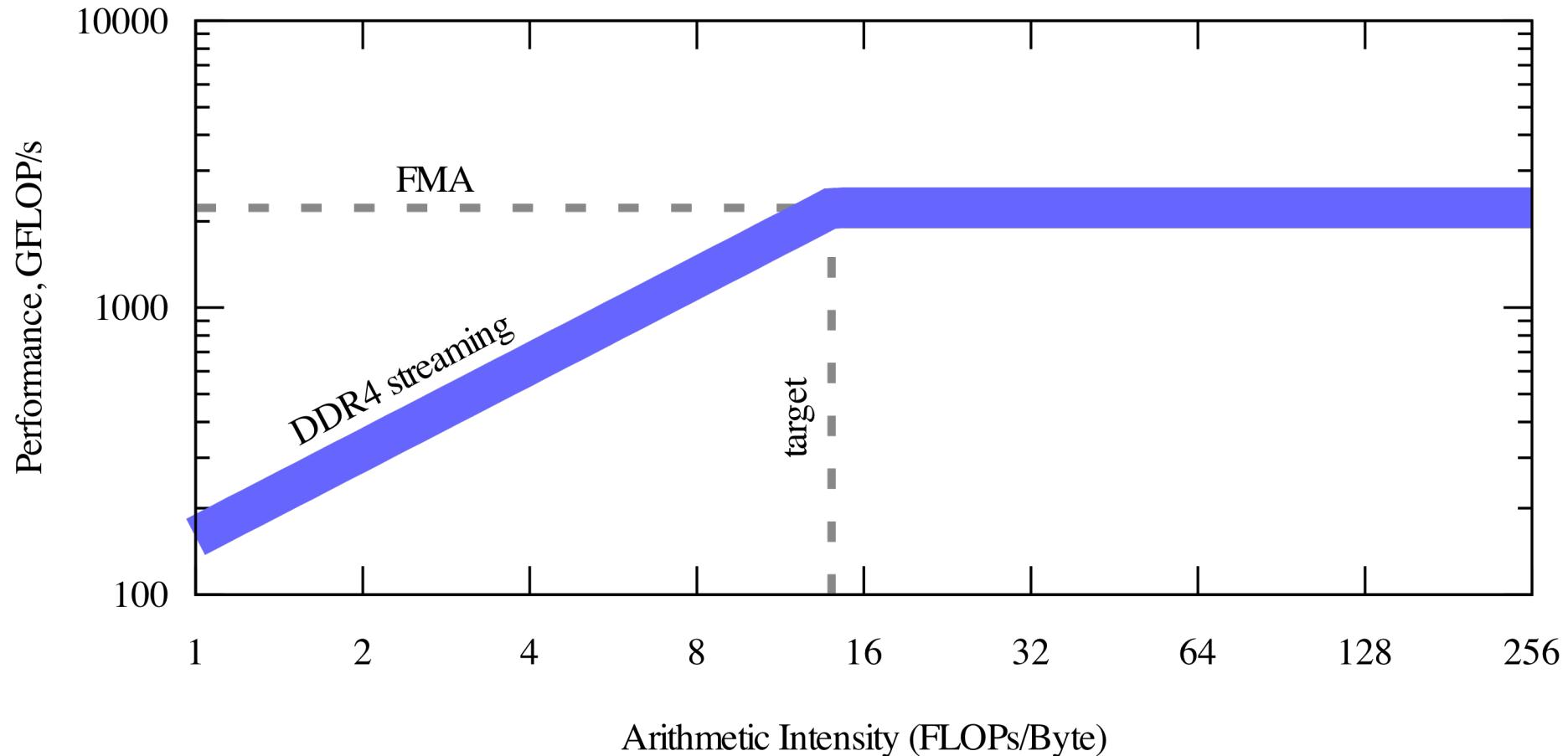
$$\begin{aligned} & 6 \text{ cores/socket} \\ & \times 2 \text{ sockets} \\ & \times 2.9 \cdot 10^9 \text{ cycles/second} \\ & \times 16 \text{ floating-point values/vector} \\ & \times 2 \text{ instructions/cycle} \\ & \times 2 \text{ FLOPs/instruction} = \\ & = 2227.2 \cdot 10^9 \text{ FLOP/s} = \mathbf{2227.2} \end{aligned}$$

GFLOP/s

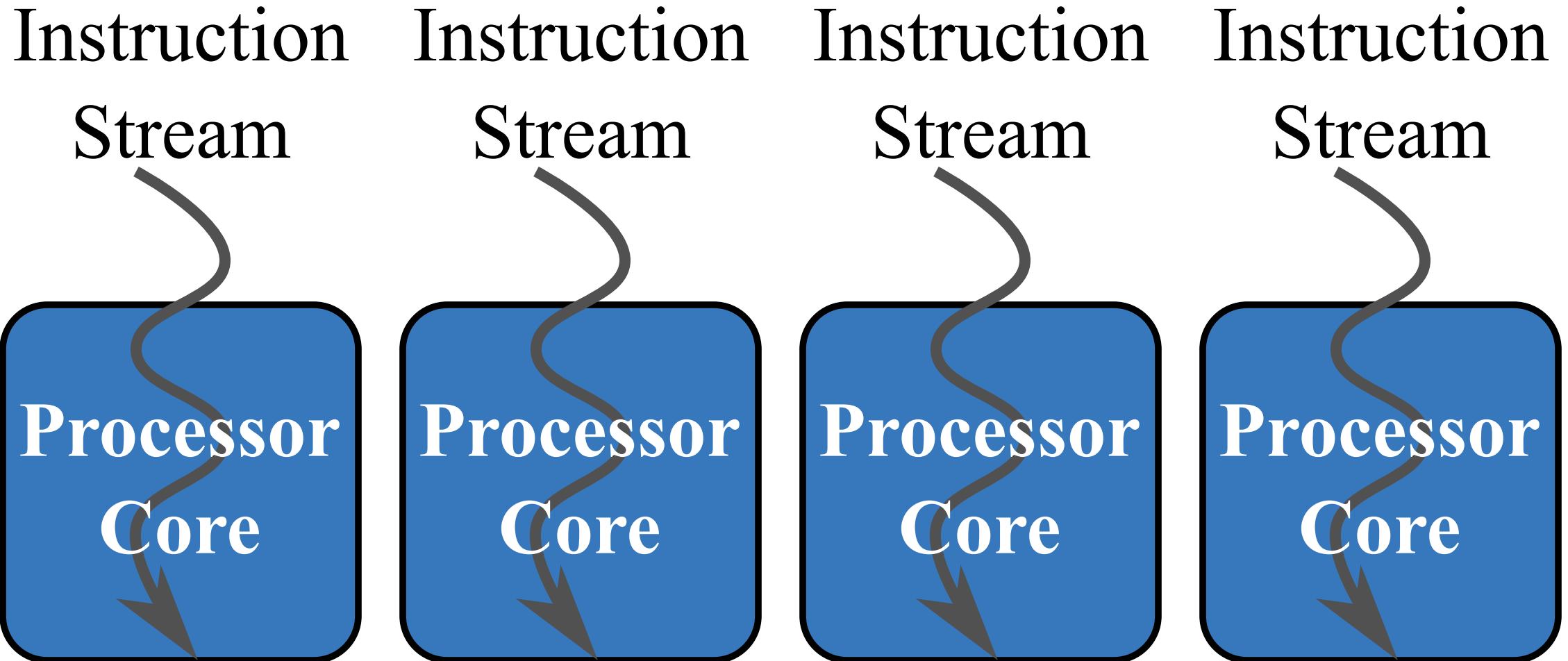
DRAM Bandwidth = **160 GB/s**

Target arithmetic Intensity = **2227.2 GFLOP/s / 160 GB/s \cong 14 FLOPs/Byte**

Roofline Diagram



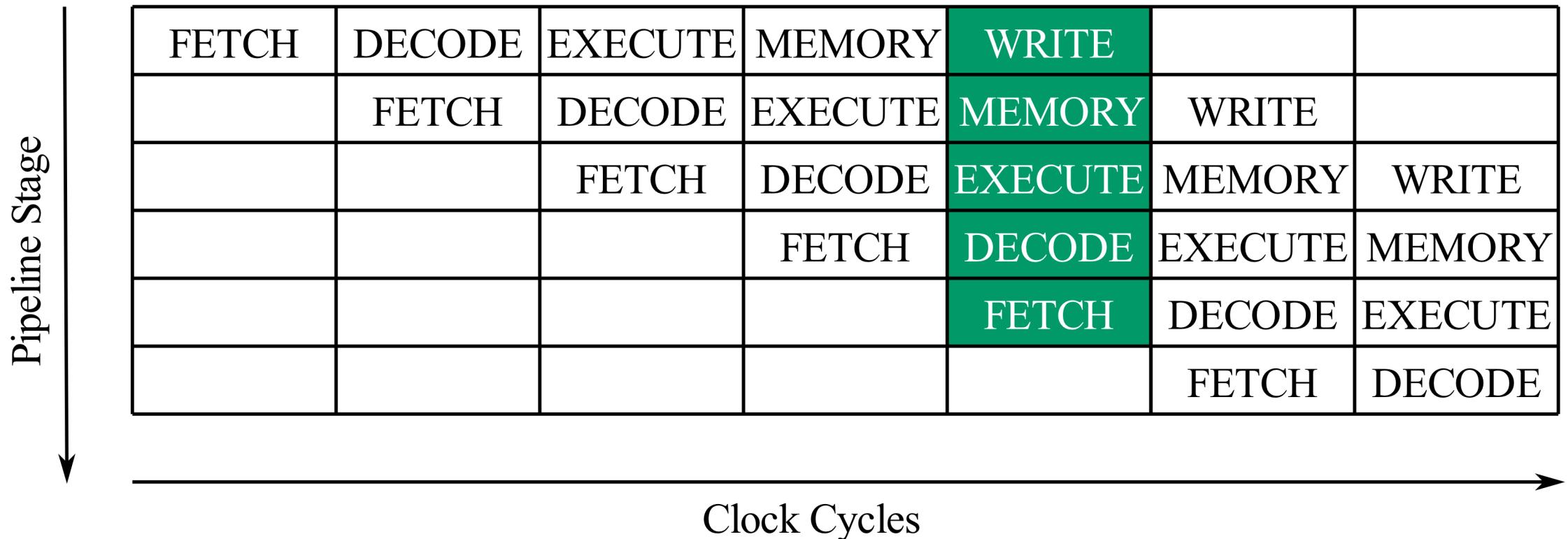
Memory Controllers and Threads



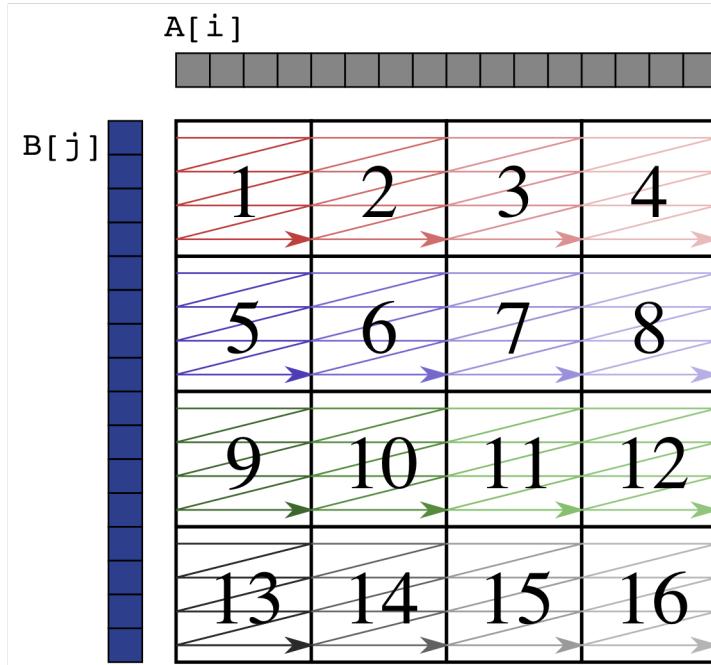
Vectorization



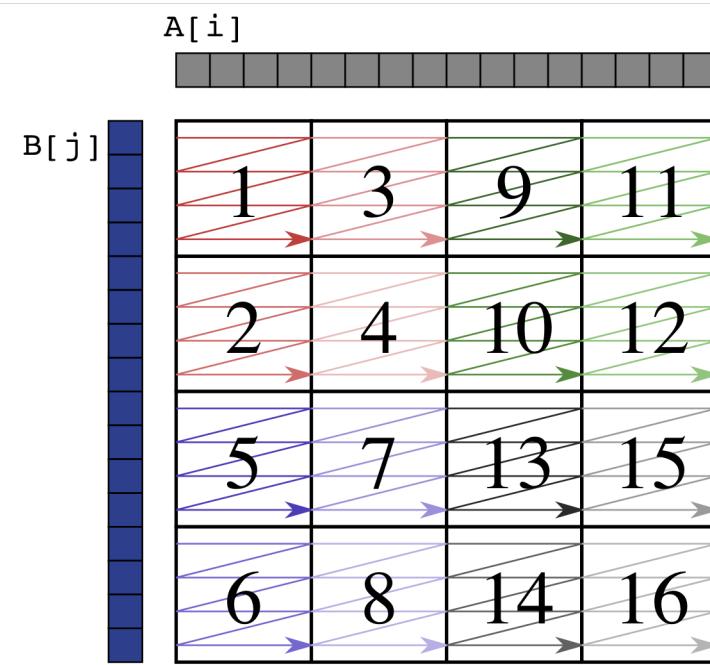
Instruction Pipelining



Data Reuse with Tiling and Recursion

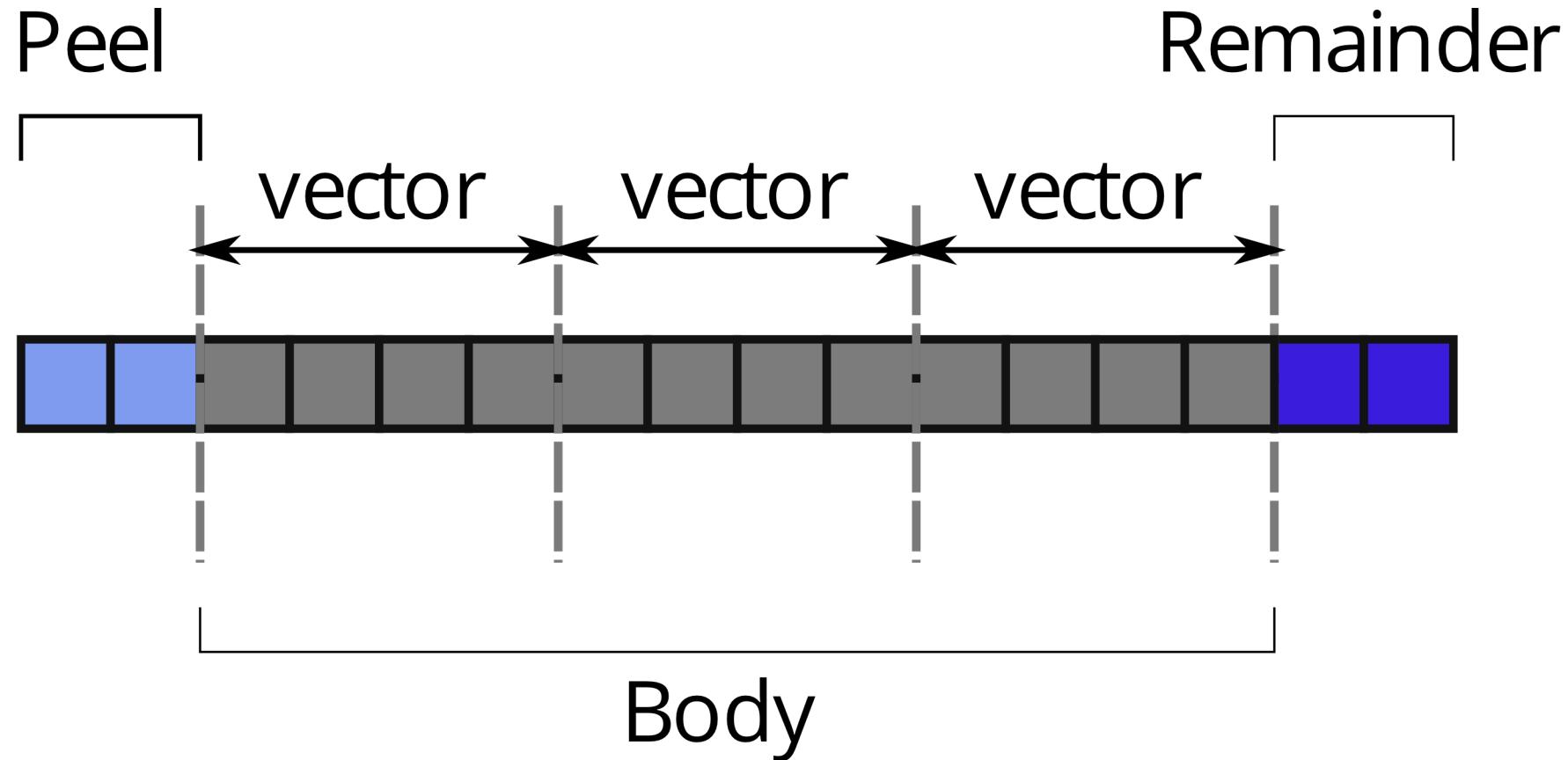


```
for (jj=0...n)
for (ii=0...n)
for (j=jj...jj+TILE)
for (i=ii...ii+TILE)
W(A[i], B[j])
```



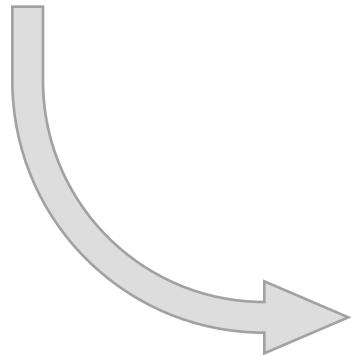
```
if (Threshold(nn,mm)) Loop();
else { if (nn > mm)
Recurse(nn/2, mm);
else
Recurse(nn, mm/2); }
```

Loop Overhead



FMACrunch: Synthetic Benchmark

```
for (v = 0; v < N_CHAINED_FMAs; v++)  
    for (l = 0; l < VECTOR_WIDTH; l++)  
        a[v*VECTOR_WIDTH + l] = a[v*VECTOR_WIDTH + l]*b[l] + c[l];
```



..B1.28:

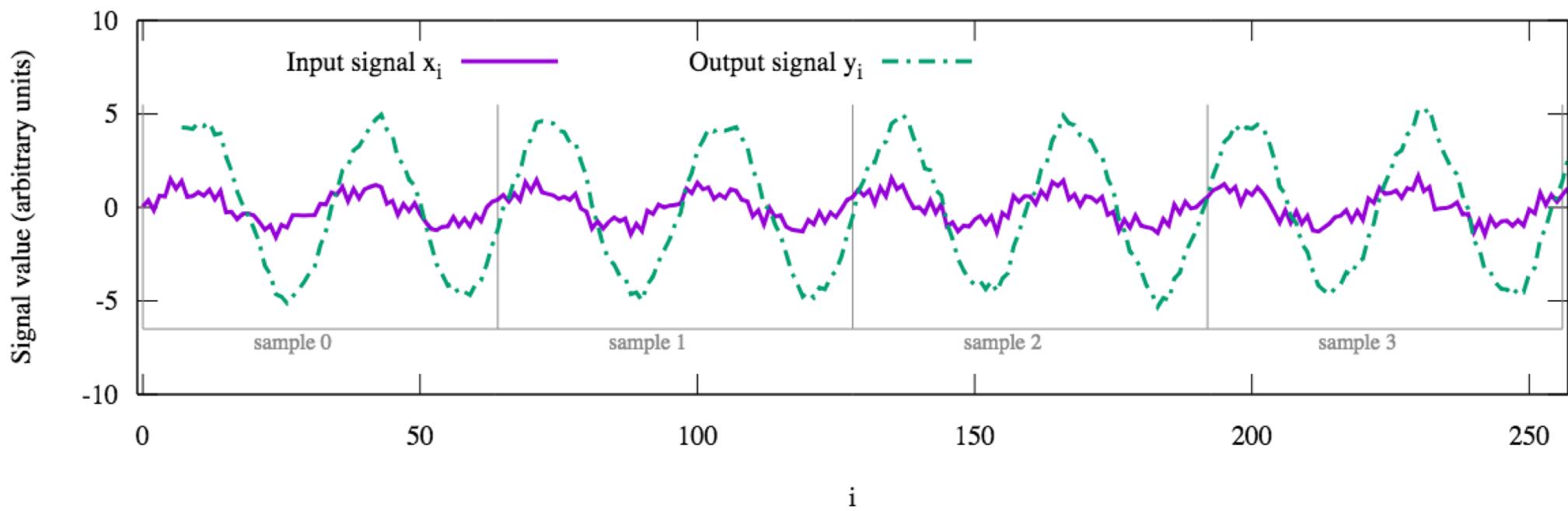
```
incq      %rdx  
vfmadd213ps %zmm8, %zmm9, %zmm7  
vfmadd213ps %zmm8, %zmm9, %zmm6  
vfmadd213ps %zmm8, %zmm9, %zmm5  
vfmadd213ps %zmm8, %zmm9, %zmm4  
vfmadd213ps %zmm8, %zmm9, %zmm3  
vfmadd213ps %zmm8, %zmm9, %zmm2  
vfmadd213ps %zmm8, %zmm9, %zmm1  
vfmadd213ps %zmm8, %zmm9, %zmm0  
cmpq      %rax, %rdx  
jb       ..B1.28      # Prob 82%
```

Access Online Materials

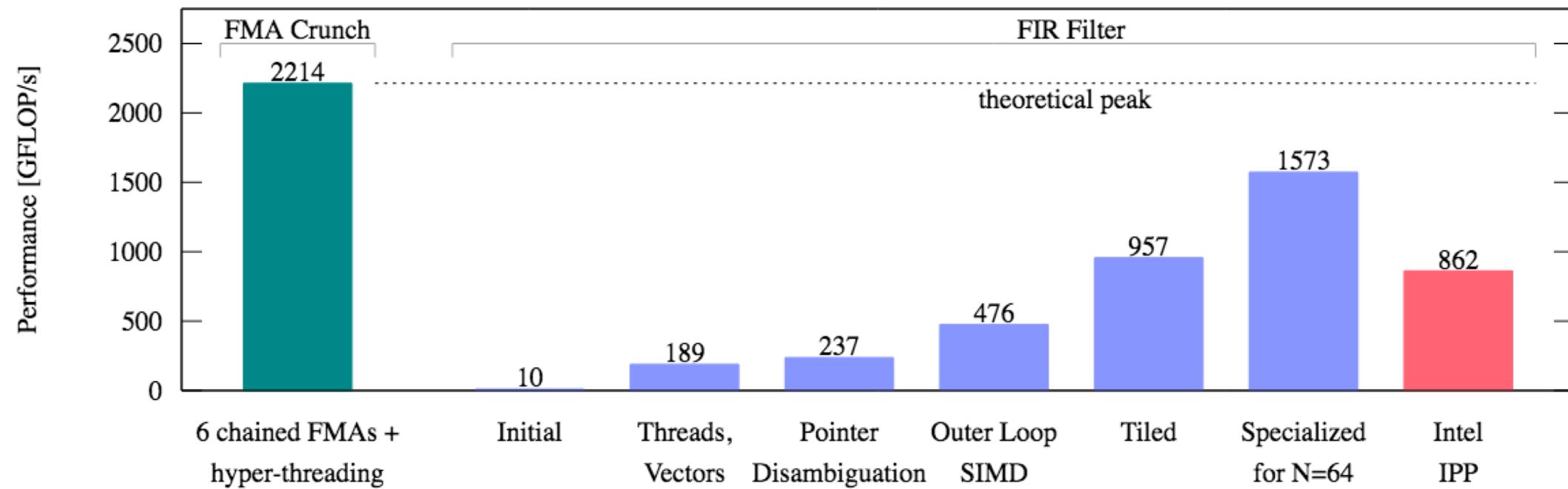
[mc2learning.com/course/
optimization-of-compute-bound-applications-c-lin/](https://mc2learning.com/course/optimization-of-compute-bound-applications-c-lin/)

FIR Filters in Computing

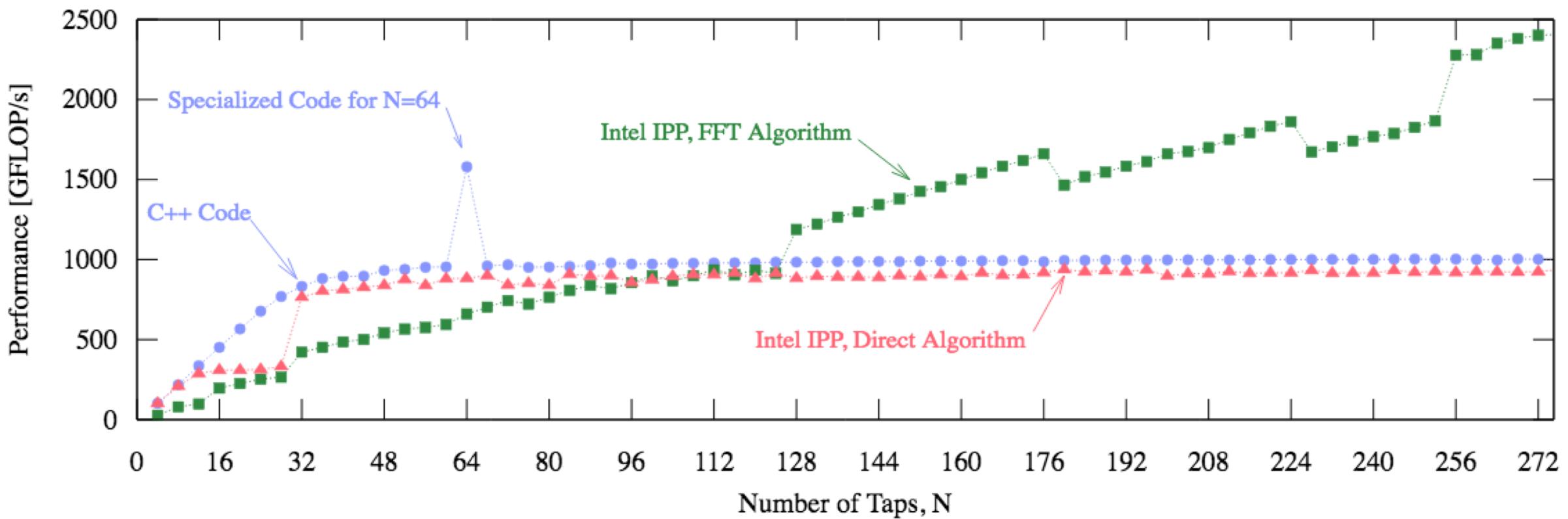
$$y_j = \sum_{k=0}^{N-1} w_k x_{j-k}$$



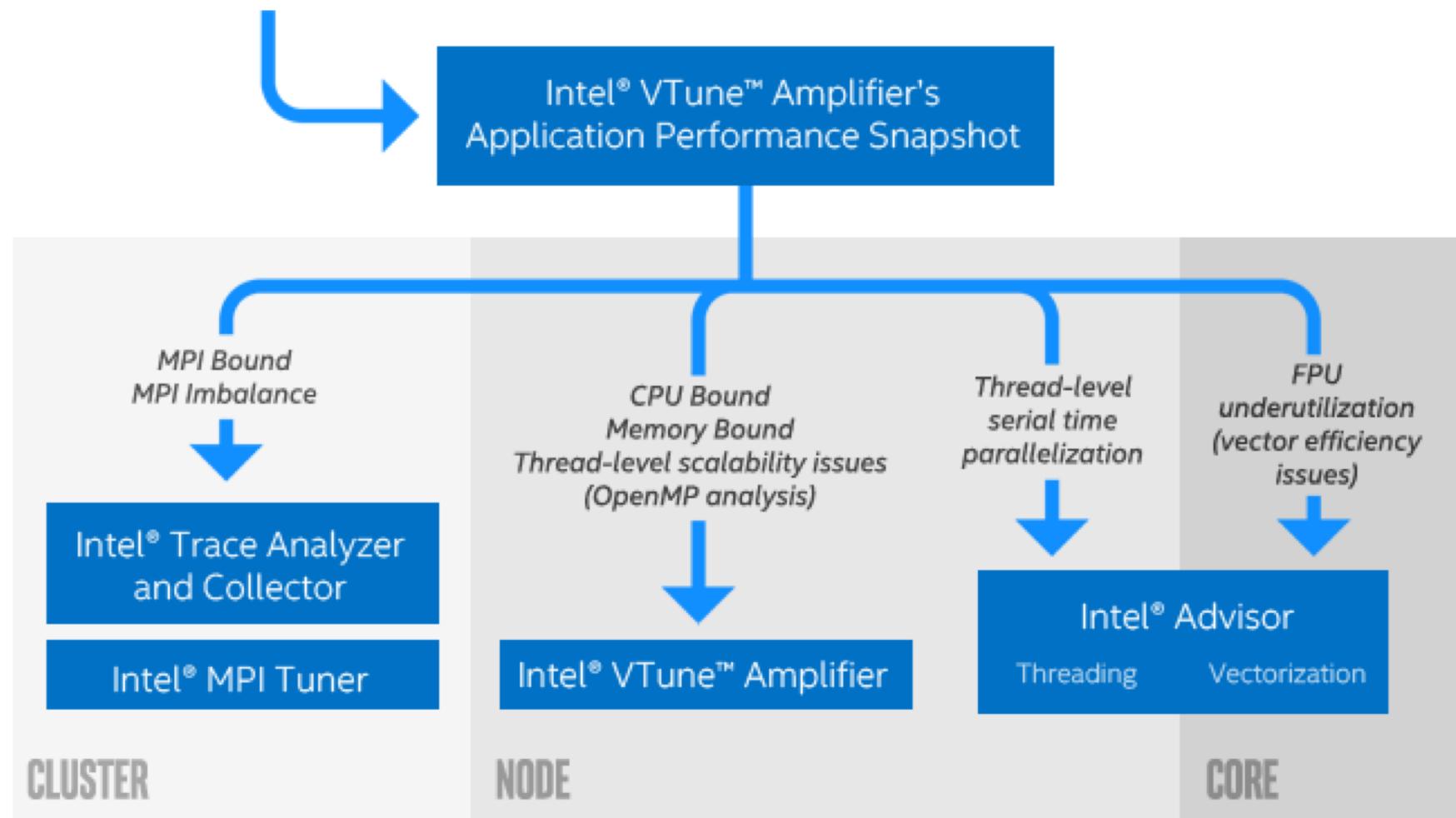
Optimization



Value Proposition of Intel IPP



Intel Parallel Studio XE Tools



Application Performance Snapshot

Intel® VTune™ Amplifier

Application Performance Snapshot

Application: emon
Report creation date: 2018-05-17 10:36:26
HW Platform: Intel(R) Xeon(R) Processor code named Skylake
Logical Core Count per node: 24
Collector type: Event-based counting driver

7.34s
Elapsed Time

276.96
SP GFLOPS

0.63
CPI

Physical Core Utilization
93.00%
Average Physical Core Utilization
11.16 out of 12.00 physical cores

Memory Stalls
4.80% of pipeline slots
Cache Stalls
16.40% of cycles
DRAM Stalls
0.30% of cycles
Average DRAM Bandwidth
11.28 GB/s
NUMA
1.80% of remote accesses

FPU Utilization
6.70%
SP.FLOPs per Cycle
4.28 Out of 64.00
Vector Capacity Usage
93.00%
FP Instruction Mix
% of Packed FP Instr.: 92.50%
% of 128-bit: 0.00%
% of 256-bit: 0.00%
% of 512-bit: 92.50%
% of Scalar FP Instr.: 7.50%
FP.Arith/Mem.Rd.Instr.Ratio
1.17
FP.Arith/Mem.Wr.Instr.Ratio
12.61

Your application underutilizes the Floating Point Unit,
either because it doesn't do much floating point math or because it is inefficient. The efficiency of both floating point and fixed point math can be improved by [optimizing vectorization](#) with tools like [Intel® Advisor](#)'s vectorization advisor.

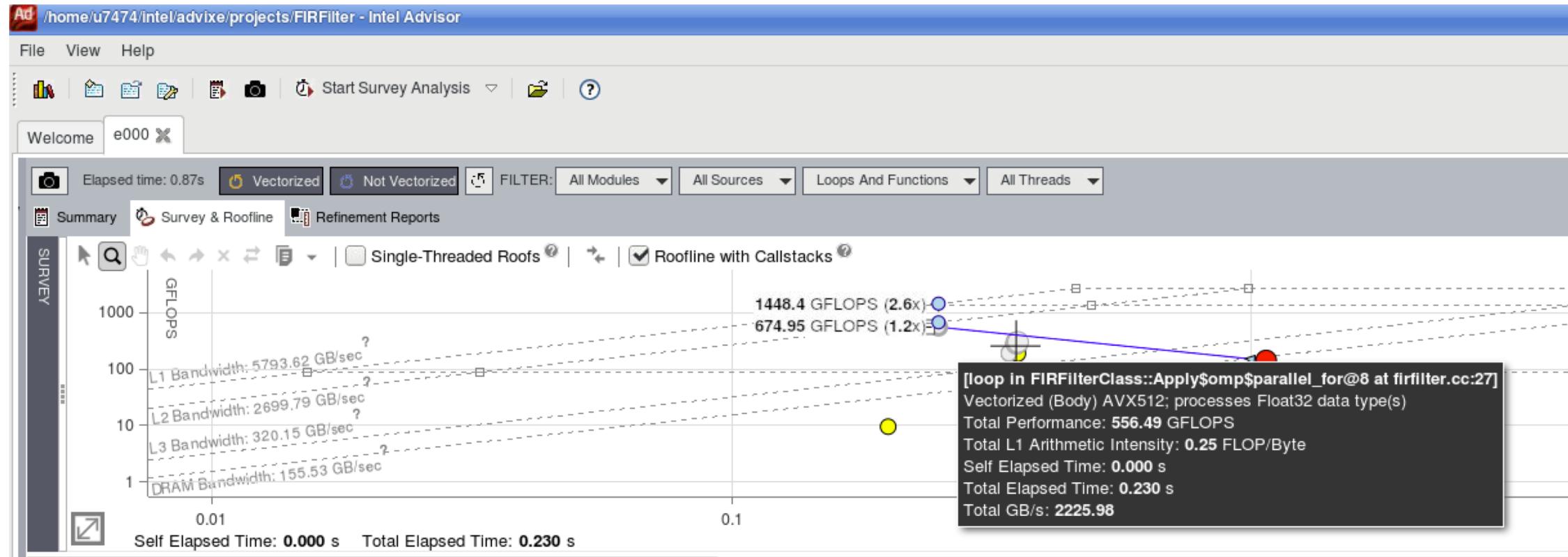
Current run Target Delta

	Current run	Target	Delta
Physical Core Utilization	93.00%	>80%	
Memory Stalls	4.80%	<20%	
FPU Utilization	6.70% 	>50%	

Resident total: 47.54 MB
Virtual total: 343.76 MB

intel

Intel Advisor XE



Intel Advisor XE

Source Top Down Code Analytics Assembly 🌟 Recommendations 🚫 Why No Vectorization?

File: firfilter.cc:27 FIRFilterClass::Apply\$omp\$parallel_for@8

Line	Source
17	
18	// Zero out the output
19	for (int j = jStart; j < L; j++) {
20	y[j] = 0;
21	}
22	
23	// Applying the FIR filter
24	for (int j = jStart; j < L; j++) {
25	
26	// Filter kernel
27	for (int k = 0; k < N; k++) {
	[loop in FIRFilterClass::Apply\$omp\$parallel_for@8 at firfilter.cc:27]
	Vectorized AVX512F_512 loop processes Float32 data type(s) and includes FMA; Permutes
	Multiversioned for data dependence, ver 1; loop was unrolled by 2
	[loop in FIRFilterClass::Apply\$omp\$parallel_for@8 at firfilter.cc:27]
	Vectorized AVX512F_512 peeled loop [not executed] processes Float32; Int32; UInt32 data type(s) and includes FMA; Mask Manipulations; Permutes
	Multiversioned for data dependence, ver 1
	[loop in FIRFilterClass::Apply\$omp\$parallel_for@8 at firfilter.cc:27]
	Vectorized AVX512F_512 remainder loop [not executed] processes Float32; Int32; UInt32 data type(s) and includes FMA; Mask Manipulations; Permutes
	Multiversioned for data dependence, ver 1
28	y[j] += weight[k]*x[j - k];
29	}
30	
31	}
32	
33	}

Intel VTune Amplifier XE

The screenshot shows the Intel VTune Amplifier XE interface with the "General Exploration" viewpoint selected. The top navigation bar includes icons for file operations and a help button, followed by the project name "r002ge" and a close button. Below the bar is a toolbar with various analysis types and a search function. The main area displays a table of assembly code metrics for the file "firfilter.cc". The table has columns for Address, Sou... Line, Assembly, Clockticks, Instructions Retired, CPI Rate, Front-End Bound, and Bad Speculation. A yellow star icon is positioned above the Clockticks column header. The first row, which contains the assembly code for "Block 33:", is highlighted with a blue background.

Address	Sou... Line	Assembly	Clockticks	Instructions Retired	CPI Rate	Front-End Bound	Bad Speculation
0x40289a	30	vbroadcastssl (%r14,%r12,4), %zmm1	455,600,000	125,800,000	3.622	0.0%	0.1%
0x4028a1	29	inc %r12	39,582,800,000	38,573,000,000	1.026	0.0%	9.3%
0x4028a4	30	vfmadd231psz (%rbx,%r8,1), %zmm1, %zmm0	0	0	0.000	0.0%	0.0%
0x4028ab	29	add \$0xfffffffffffffff, %rbx	138,097,800,000	205,380,400,000	0.672	0.0%	0.0%
0x4028af	29	cmp %r9, %r12	37,400,000	13,600,000	2.750	0.0%	0.0%
0x4028b2	29	jb 0x40289a <Block 33>					