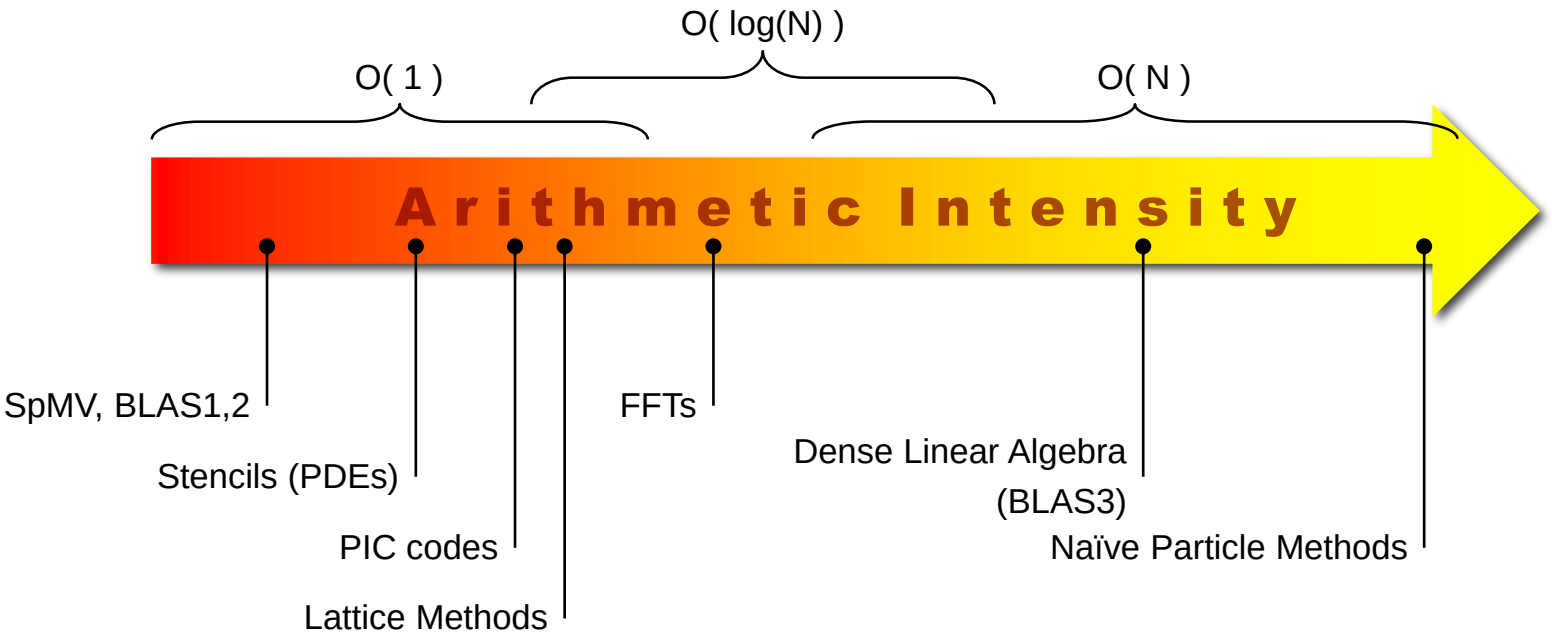
# Roofline Model

Slide Courtesy: S. Williams, Xiaoye Sherry Li and others

Slides adapted by: Sparsh Mittal

# Arithmetic Intensity



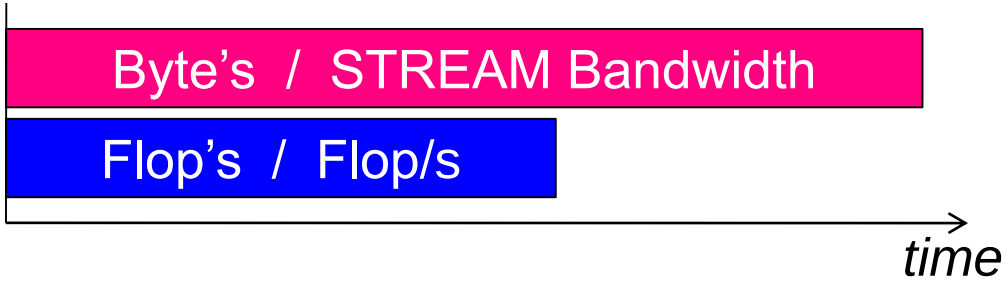

- Arithmetic Intensity (AI) ~ Total Flops / Total DRAM Bytes
  - E.g.: dense matrix-matrix multiplication: $n^3$ flops / $n^2$ memory

- Higher AI → better locality → amenable to many optimizations → achieve higher % machine peak

# Roofline model: Basic concept

- Synthesize communication, computation, and locality into a single visually-intuitive performance figure using bound and bottleneck analysis.
  - Assume FP kernel maintained in DRAM, and perfectly overlap computation and communication w/ DRAM
  - Arithmetic Intensity (AI) is computed based on DRAM traffic after being filtered by cache
- Question : is the code computation-bound or memory-bound?

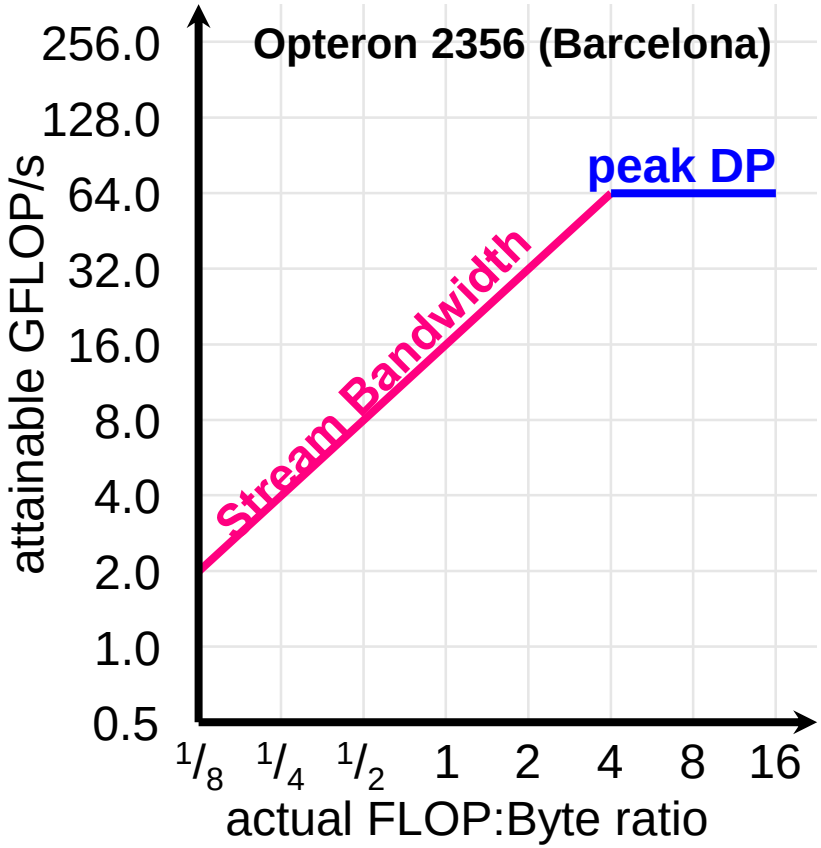
- Time is the maximum of the time required to transfer the data and the time required to perform the floating point operations.



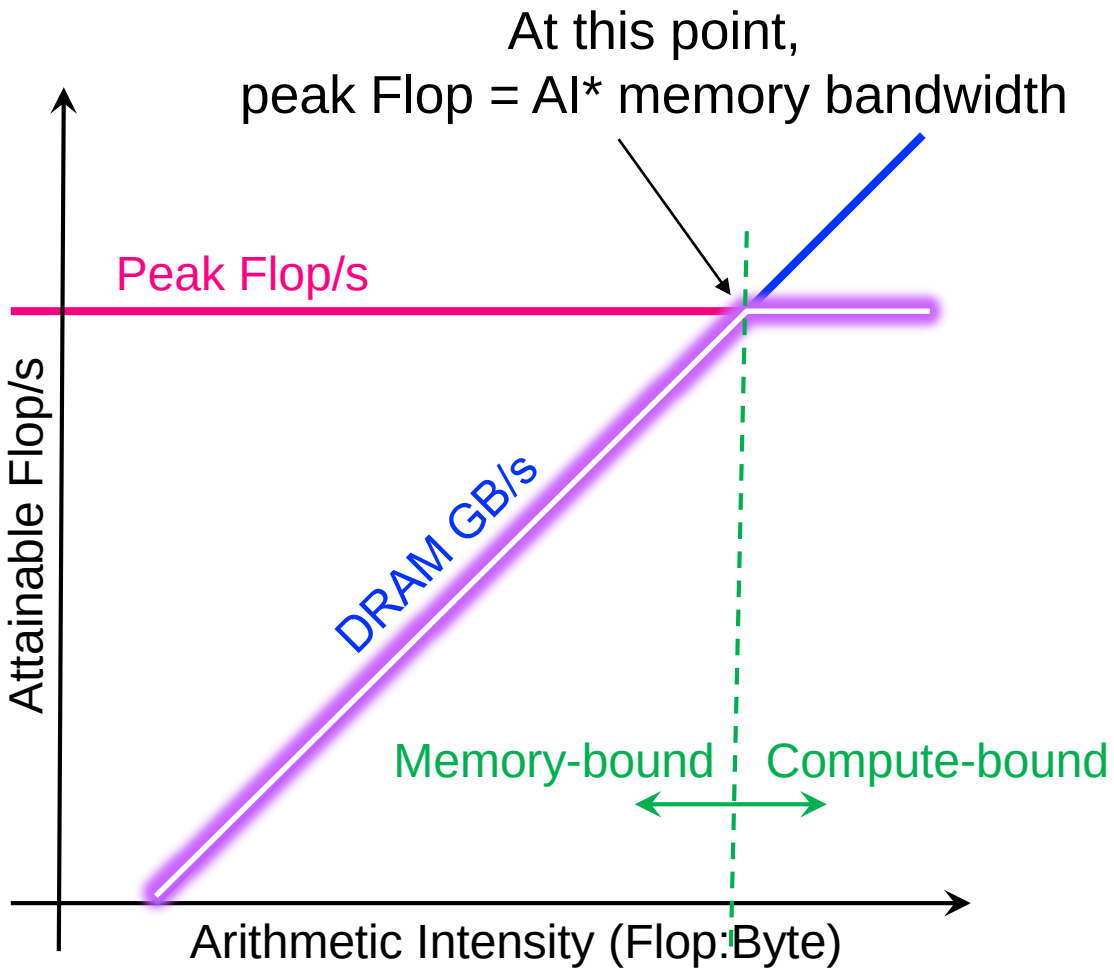
# Roofline model

$$\text{Attainable Performance}_{ij} = \min \begin{cases} \text{FLOP/s} \ (\text{with Optimizations}_{1\text{-}i}) \\ \text{AI} * \text{Bandwidth} \ (\text{with Optimizations}_{1\text{-}j}) \end{cases}$$

- Roofline
  - Given the code AI, can inspect the figure to bound performance
  - Provides insights as to which optimizations will potentially be beneficial
- Machine-dependent, code-dependent

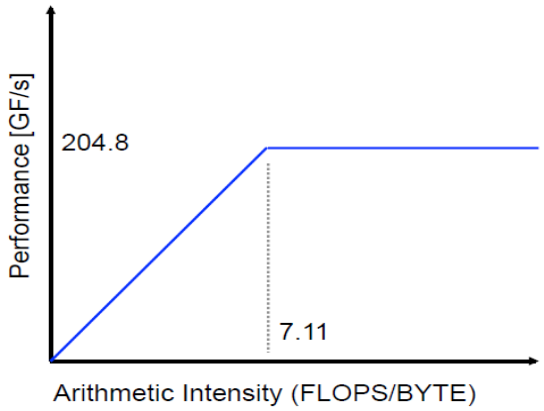


# Insights


At this point,
peak Flop = AI* memory bandwidth
Peak Flop/s
DRAM GB/s
Attainable Flop/s
Memory-bound
Compute-bound
Arithmetic Intensity (Flop:Byte)

# Example

- Consider a roofline model, whose ridge point has (x, y) coordinates as (7.11,204.8). Find the performance for AI = 1/14. Also, tell whether we are in bandwidth-bound or compute-bound region.

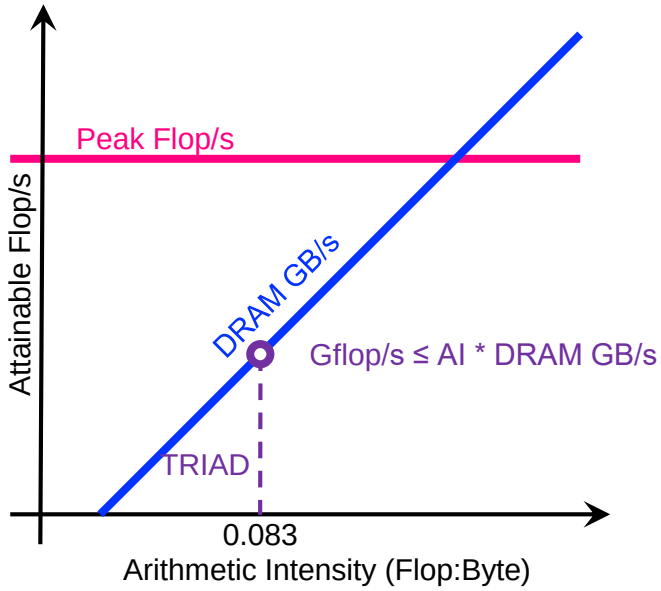



- Since AI < 7.11, we are in memory bandwidth-bound region.
- At ridge point: AI*bandwidth = PeakFLOP
- 7.11* bandwidth = 204.8
- => bandwidth = 28.8
- Now, for AI = 1/14, we get performance as AI* bandwidth = 28.8/14 = 2.057

# Roofline Example #1: STREAM Triad

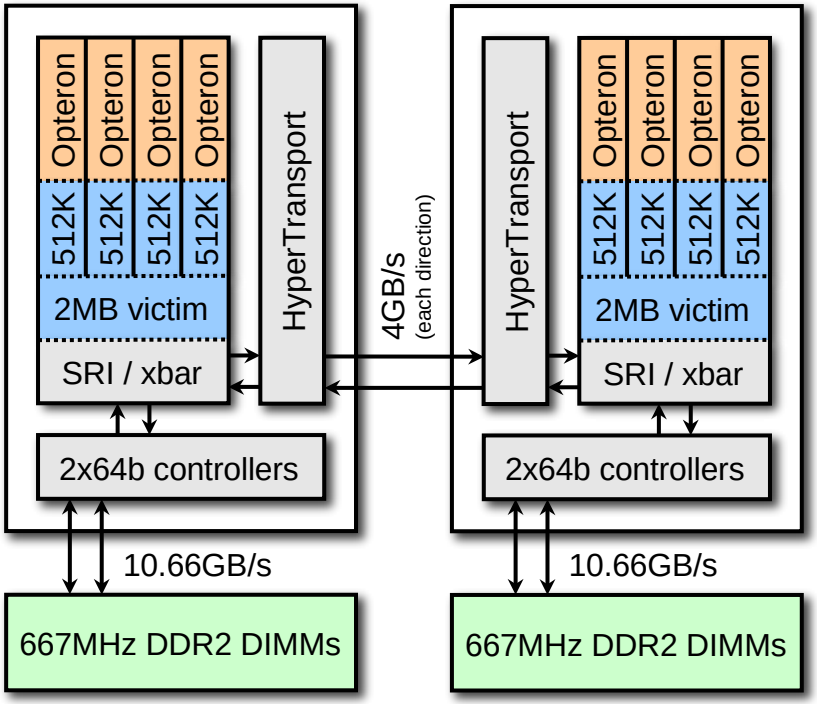
```
for(i=0;i<N;i++){
  Z[i] = X[i] + alpha*Y[i];
}
```

- 2 flops per iteration
- Transfer 24 bytes per iteration (read X[i], Y[i], write Z[i])
- **AI = 0.083 flops per byte == Memory bound**



# Example of Opteron Processor

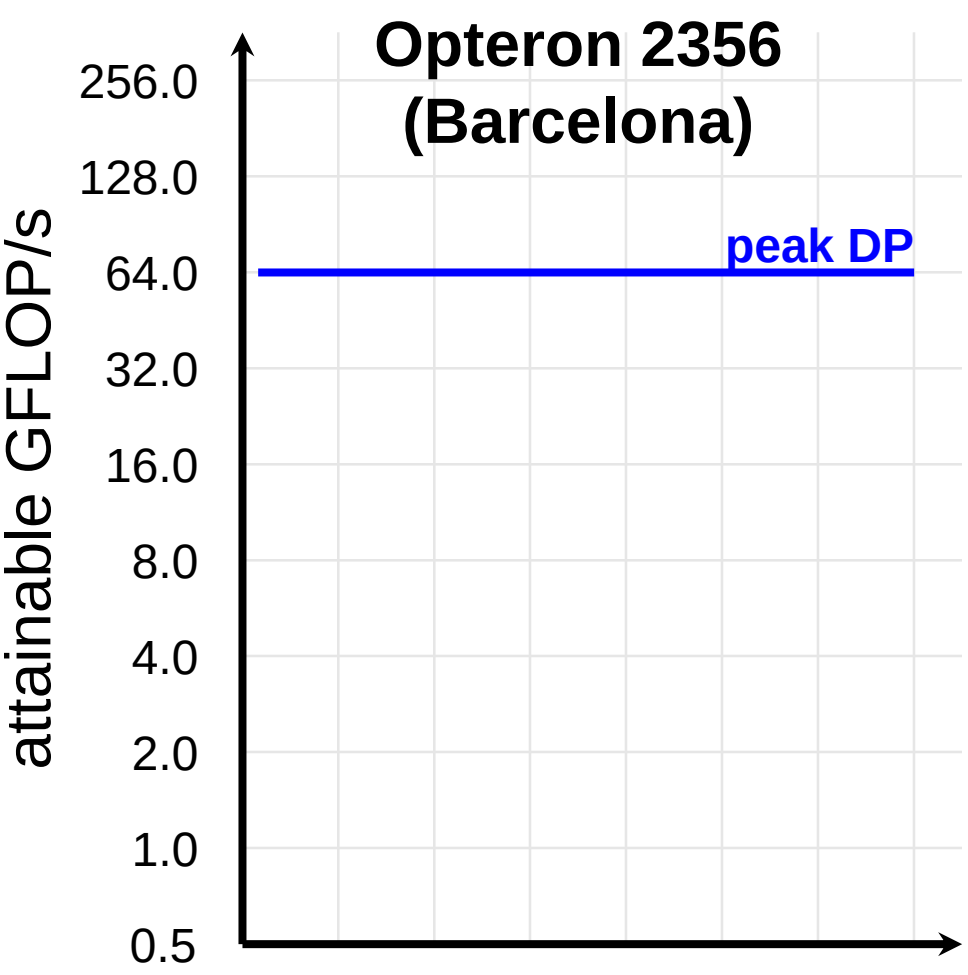
- Consider the Opteron 2356:
  - Dual Socket (NUMA)
  - limited HW stream prefetchers
  - quad-core (8 total)
  - 2.3GHz
  - 2-way SIMD (DP)
  - separate FPMUL and FPADD datapaths
  - 4-cycle FP latency




- Let us study its roofline model

# Roofline Model

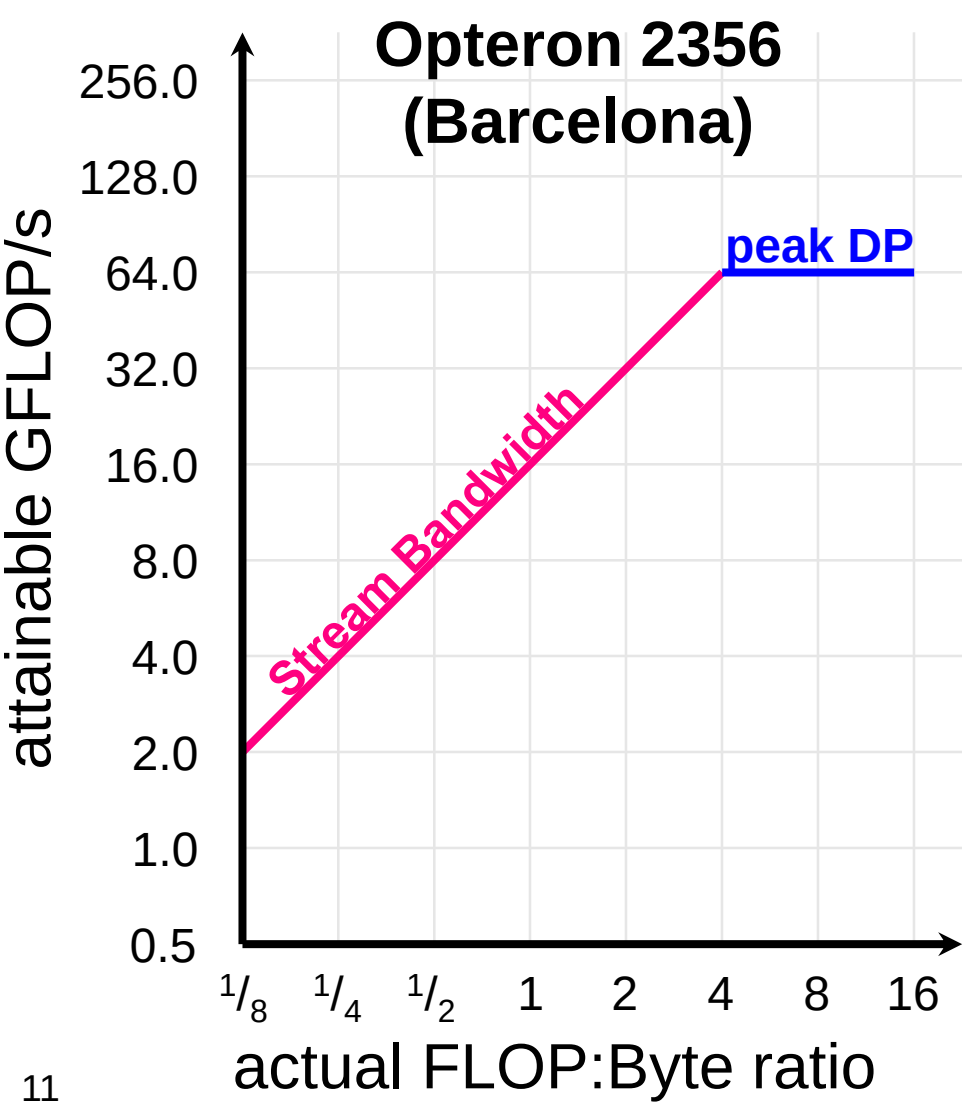
## Basic Concept




- Naively, one might assume peak performance is always attainable.

# Roofline Model

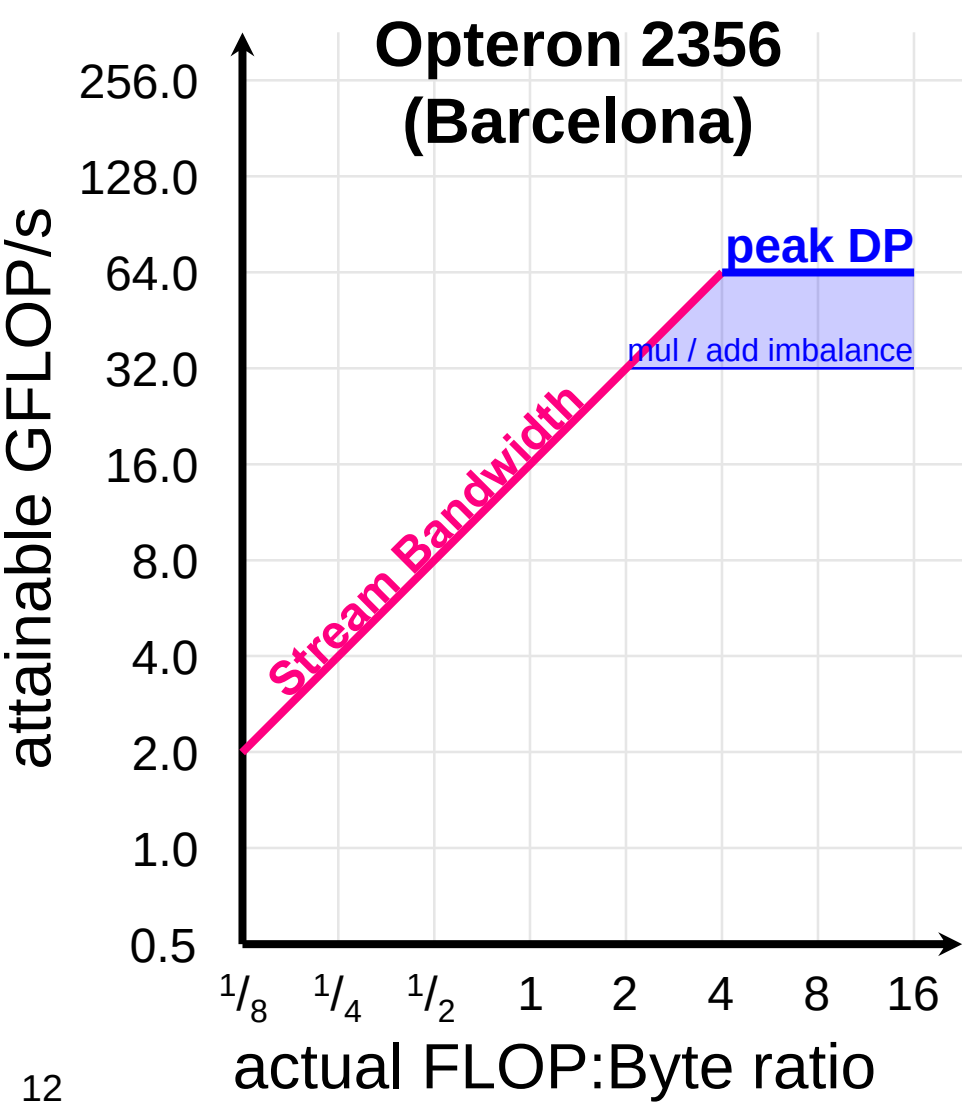
## Basic Concept




- However, with a lack of locality, **DRAM bandwidth can be a bottleneck**
- Plot on log-log scale
- Given AI, we can easily bound performance
- But architectures are much more complicated
- We will bound performance as we eliminate specific forms of in-core parallelism

# Roofline Model

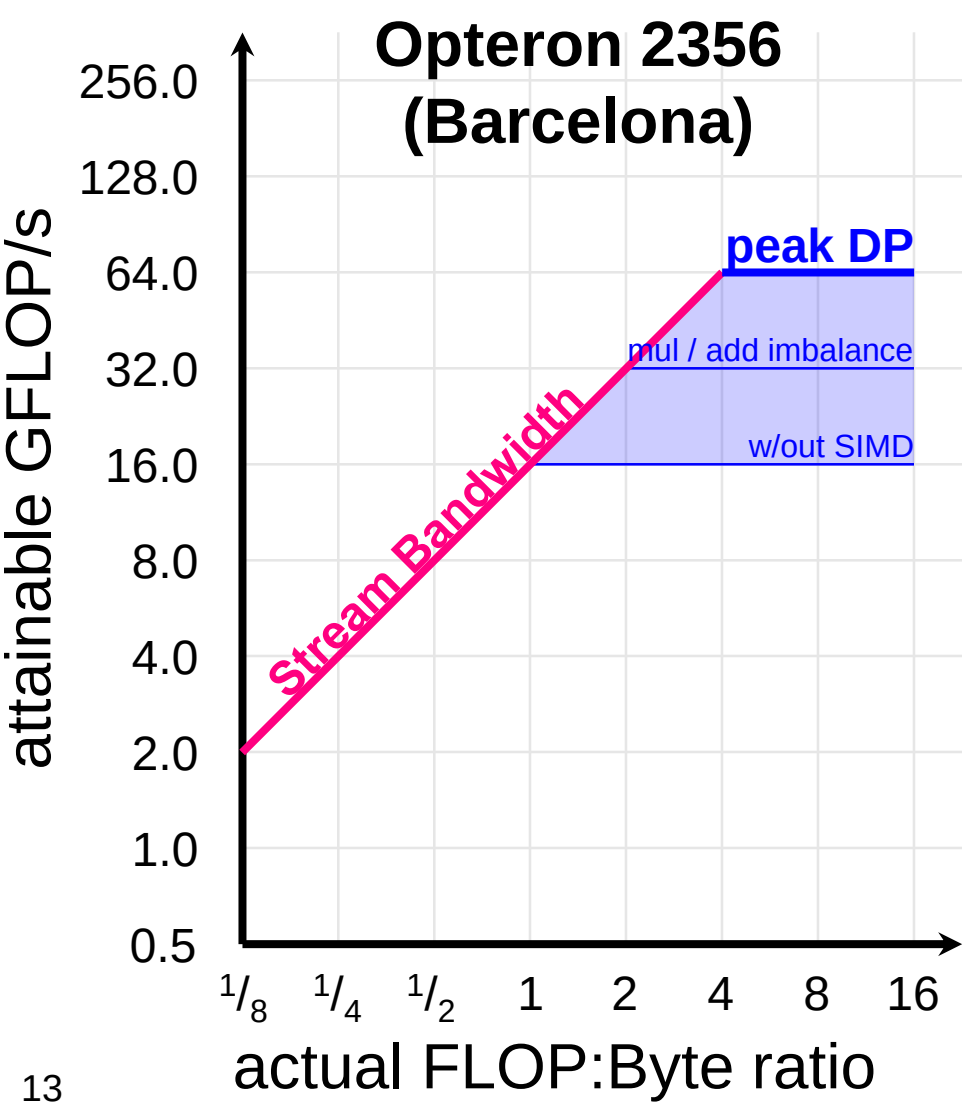
## computational ceilings




- Opterons have dedicated multipliers and adders.
- If the code is dominated by adds, then attainable performance is half of peak.
- We call these **Ceilings**
- They act like constraints on performance

# Roofline Model

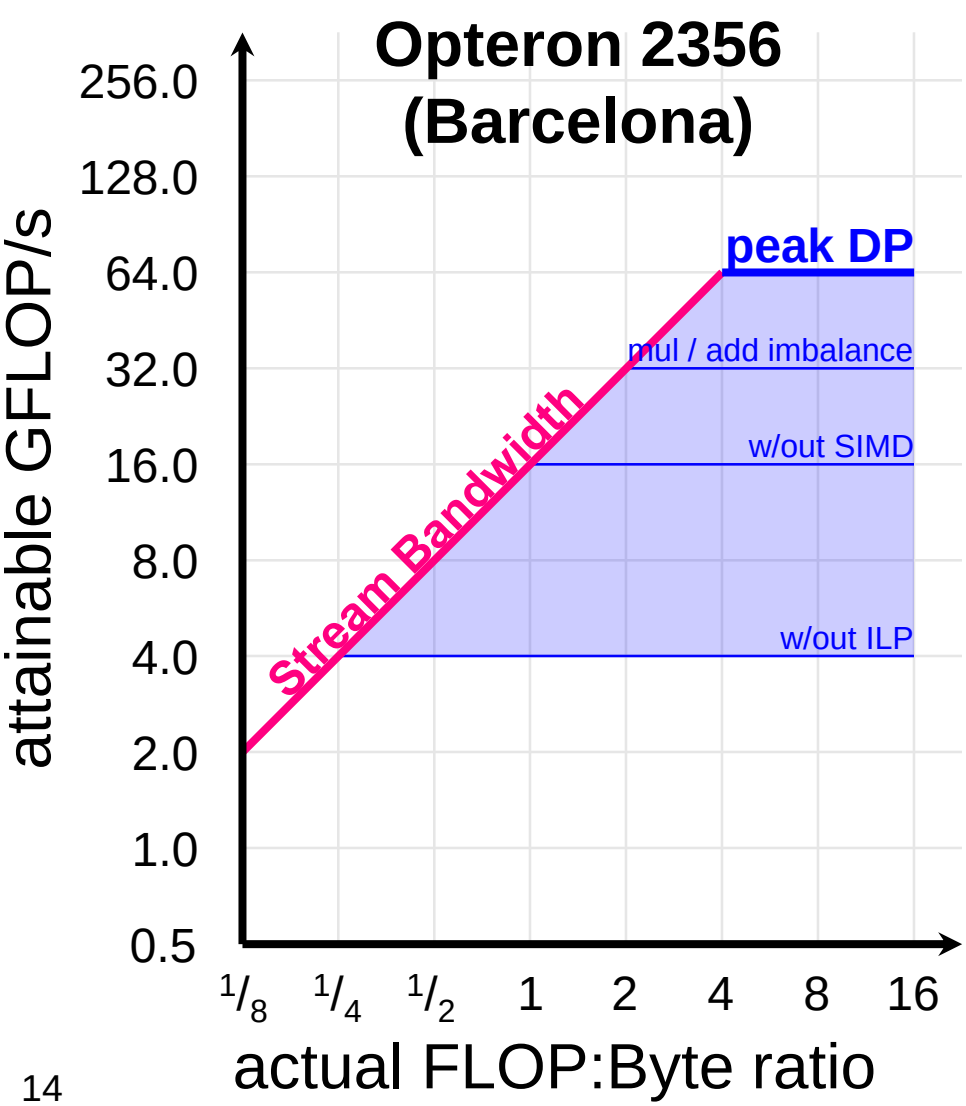
## computational ceilings




- Opterons have 128-bit datapaths.
- If instructions aren't SIMDized, attainable performance will be halved

# Roofline Model

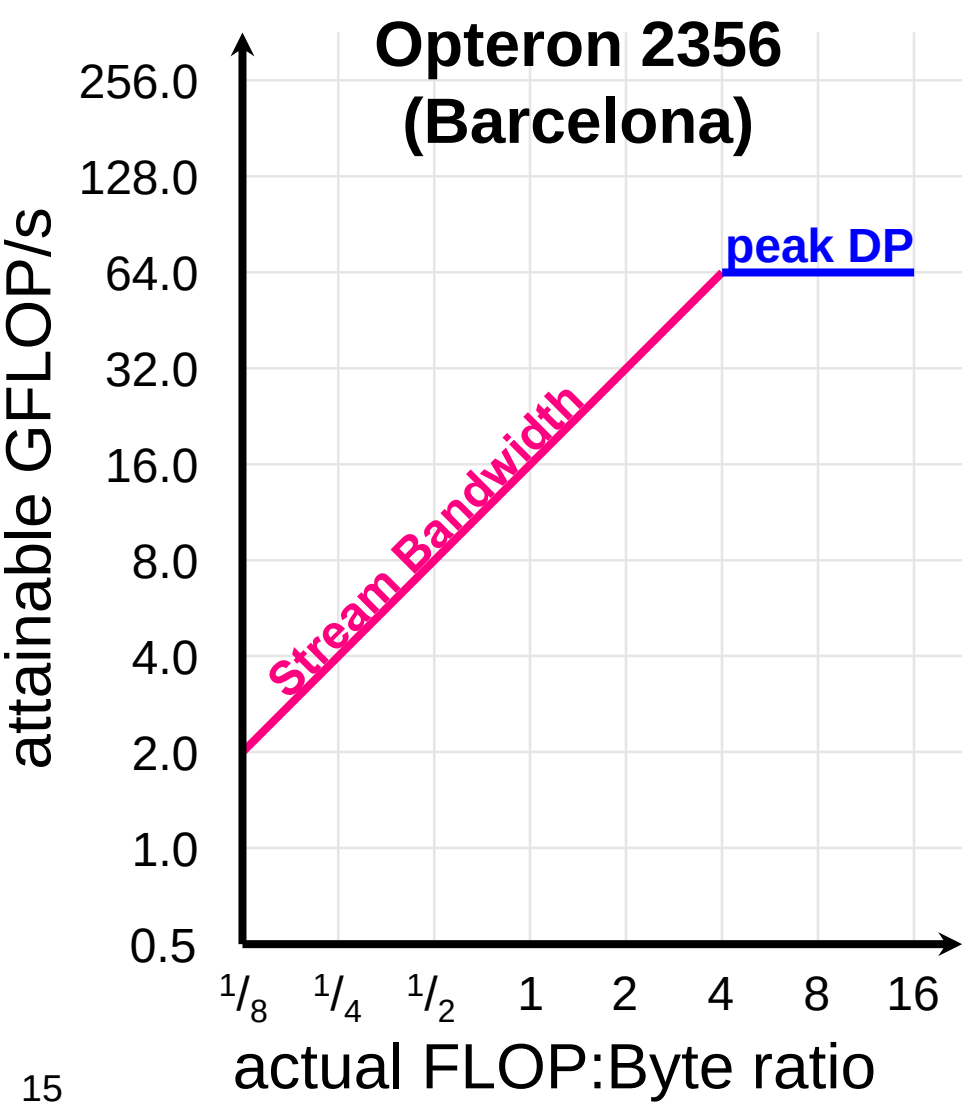
## computational ceilings




- On Opterons, floating-point instructions have a 4 cycle latency.
- If we don't express 4-way ILP, performance will drop by as much as 4x

# Roofline Model

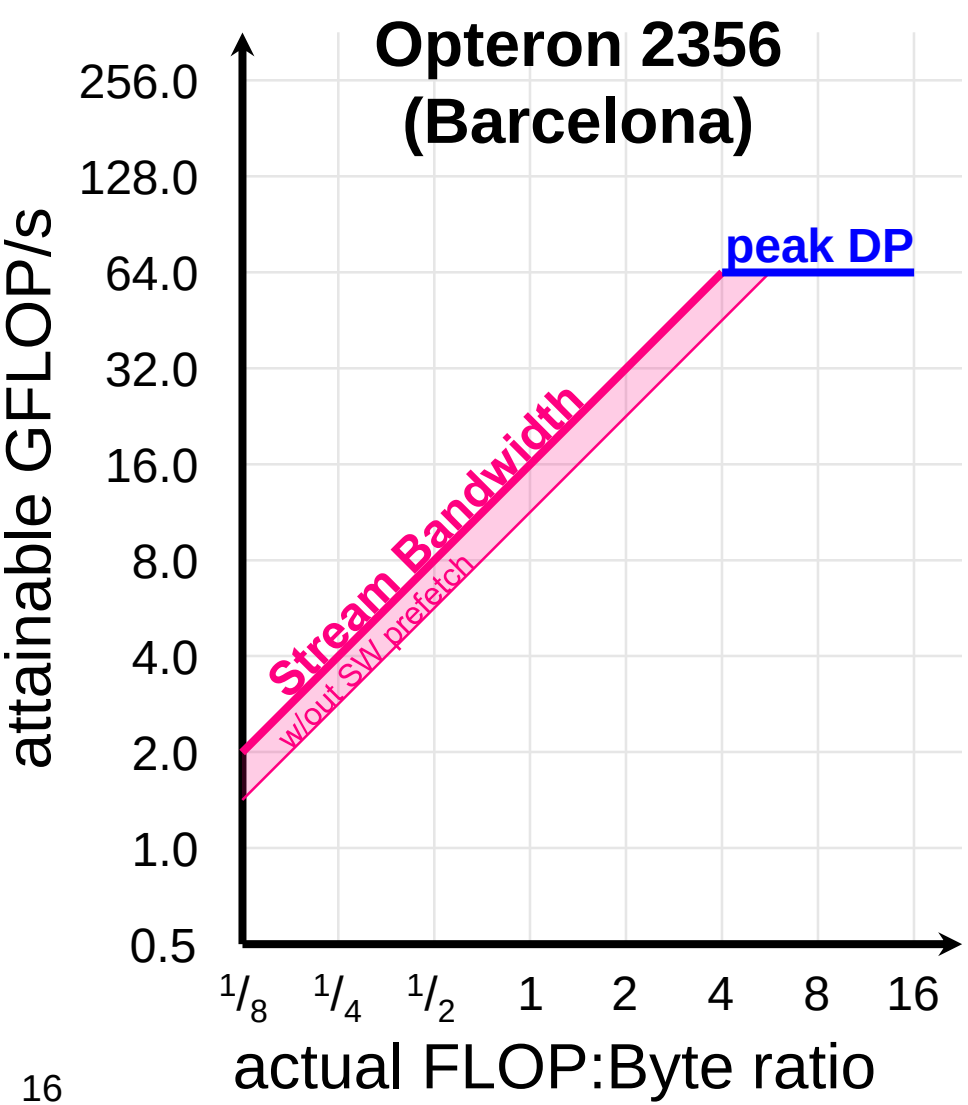
## communication ceilings




- We can perform a similar exercise taking away parallelism from the memory subsystem

# Roofline Model

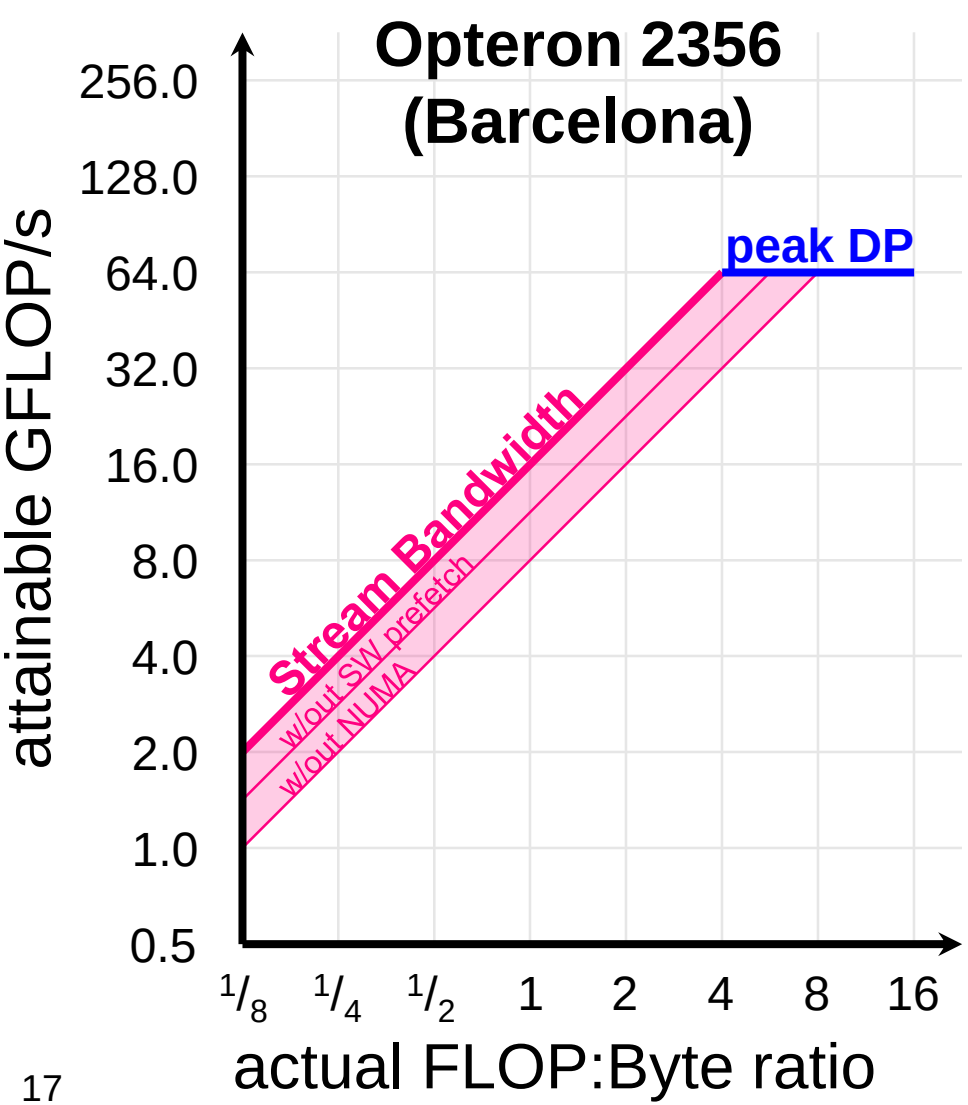
## communication ceilings




- Explicit software prefetch instructions are required to achieve peak bandwidth

# Roofline Model

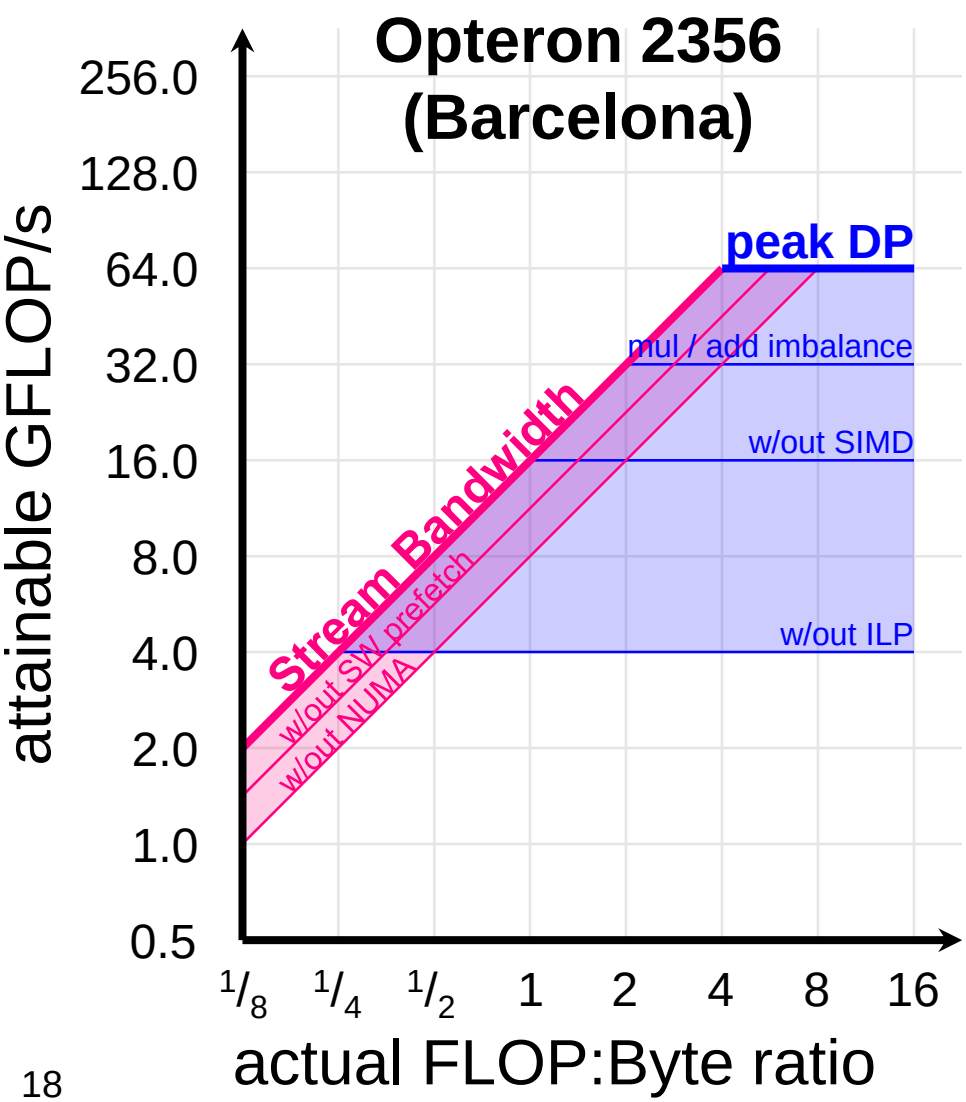
## communication ceilings




- Opterons are NUMA
- As such memory traffic must be correctly balanced among the two sockets to achieve good Stream bandwidth.
- We could continue this by examining strided or random memory access patterns

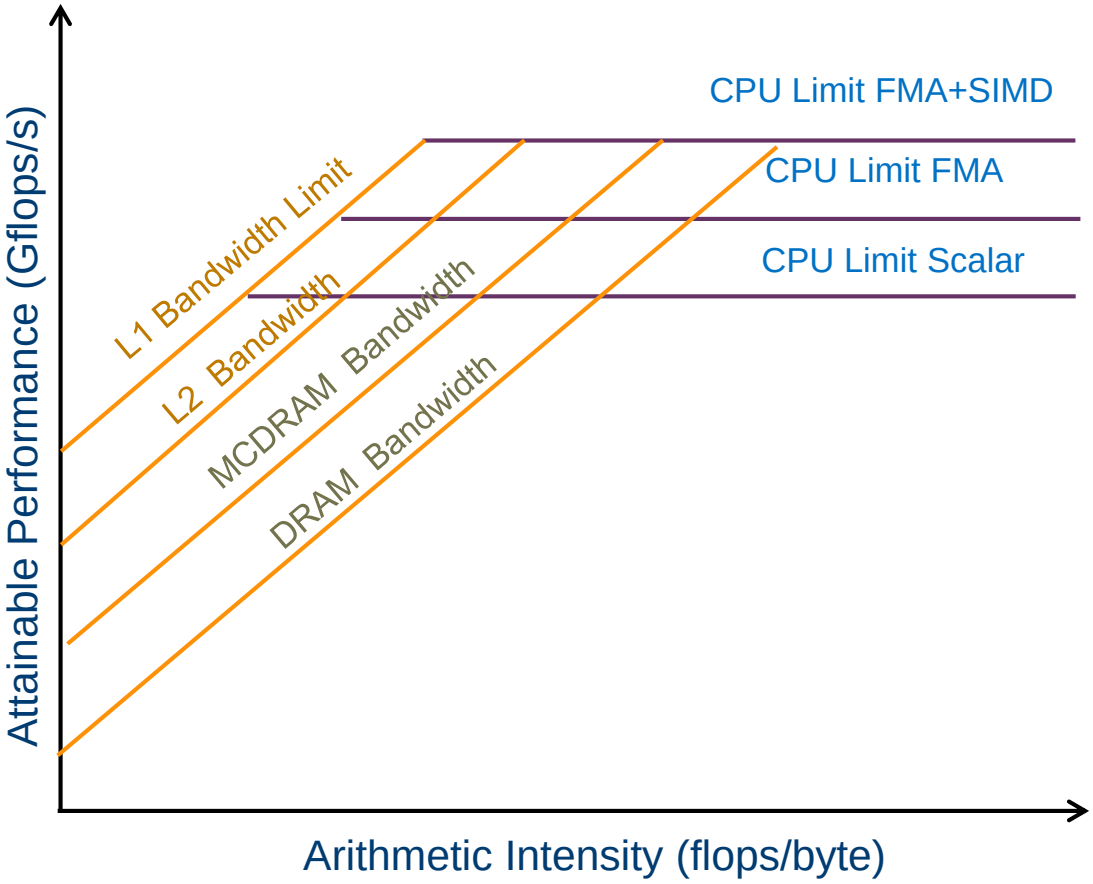
# Roofline Model

## computation + communication ceilings




- We may bound performance based on the combination of expressed in-core parallelism and attained bandwidth.

# There Are Different Ceilings




KNL theoretical peak performance = AVX Frequency (1.4 Ghz) x 8 (vector width) x 2 (dual vpus) x 2 (FMA inst.) x 68 (number of Cores) = 3.0 TFlops/s

In reality mileage may vary, it's usually best to measure peak bandwidths and peak performance by micro benchmarks

# Numerical Example

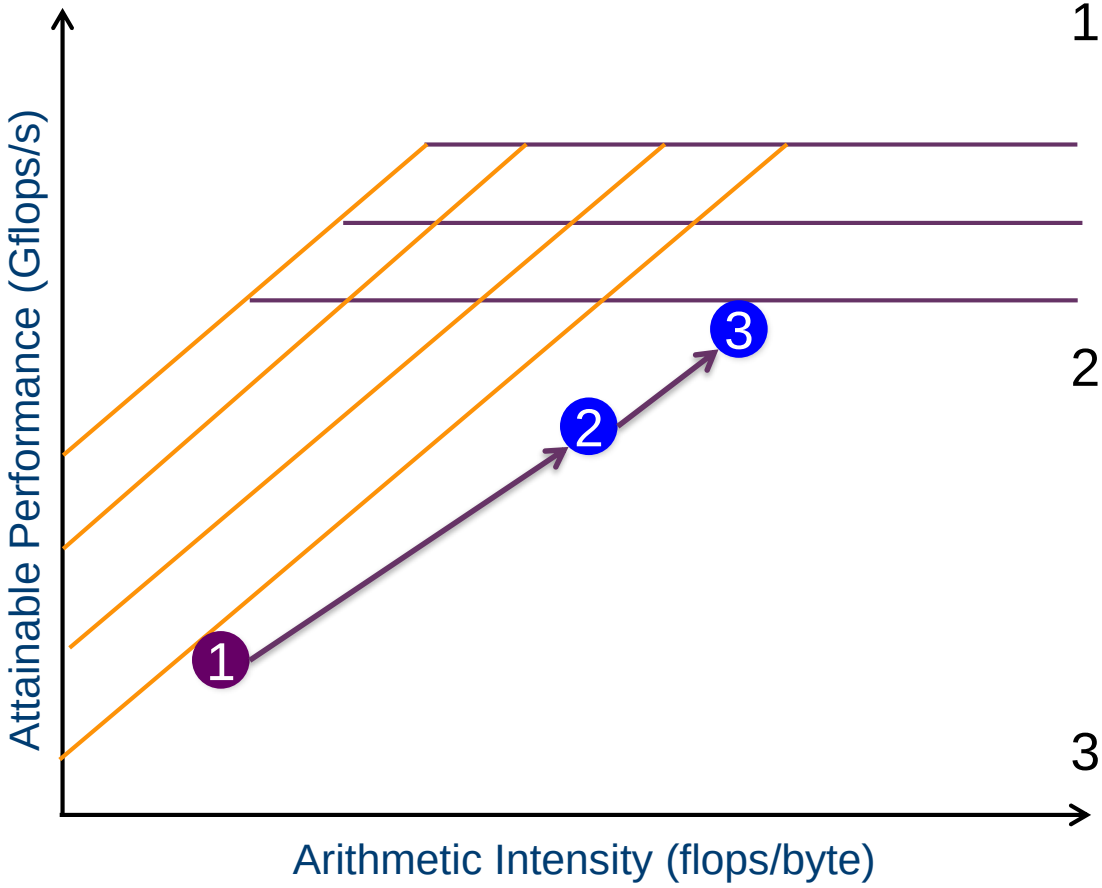
- Xeon Phi 7250 has 3.05 TFLOPS peak DP
- 1.4 GHz × 68 core × 8 SIMD × 2 VPUs × 2 FMA = 3046.4 GFLOPS

- AVX frequency is only 1.2 GHz and might throttle down under heavy load
- ⇒ actual peak: 2611.2 GFLOPS

- Add more FLOPS ceilings
- Without instruction level parallelism (ILP), i.e. dual VPUs and FMA
  1.2 GHz × 68 core × 8 SIMD = 652.8 GFLOPS

- without ILP, and without SIMD
- 1.2 GHz × 68 core = 84.6 GFLOPS

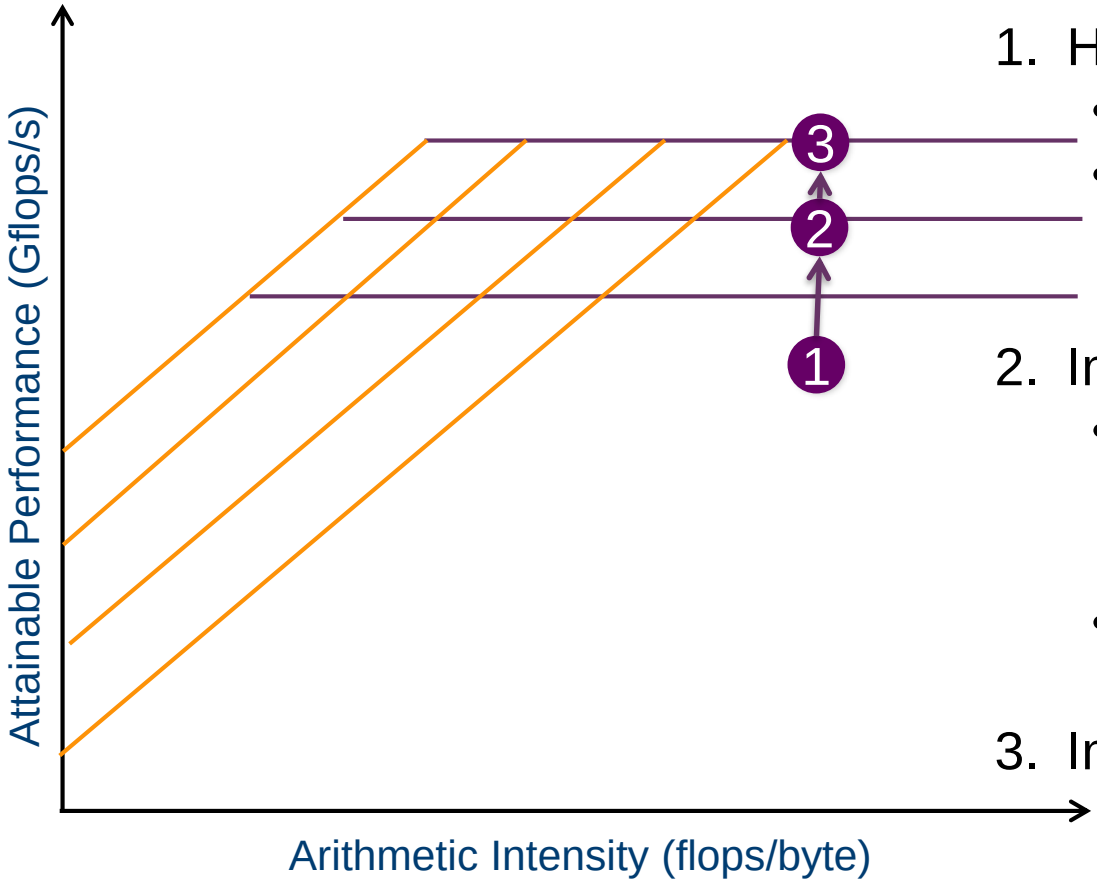
# Example 1: Memory Bound Application




1. Low Ai, “STREAM-like” application. Assume it’s well vectorized
   - No cache reuse
     → DRAM bandwidth bound
     → DRAM AI = L1 AI
2. Implement L2 cache optimization
   - L2 Cache is fully reused, GFLOPS increase
     → roofline moves to the right because we are doing less loads from DRAM.
3. Implement L1 cache optimization
   - See 2.
     roofline is bound by the scalar add peak

[1] S. Williams et al. *CACM* (2009), crd.lbl.gov/departments/computer-science/PAR/research/roofline

# Example 2: Compute Bound Application

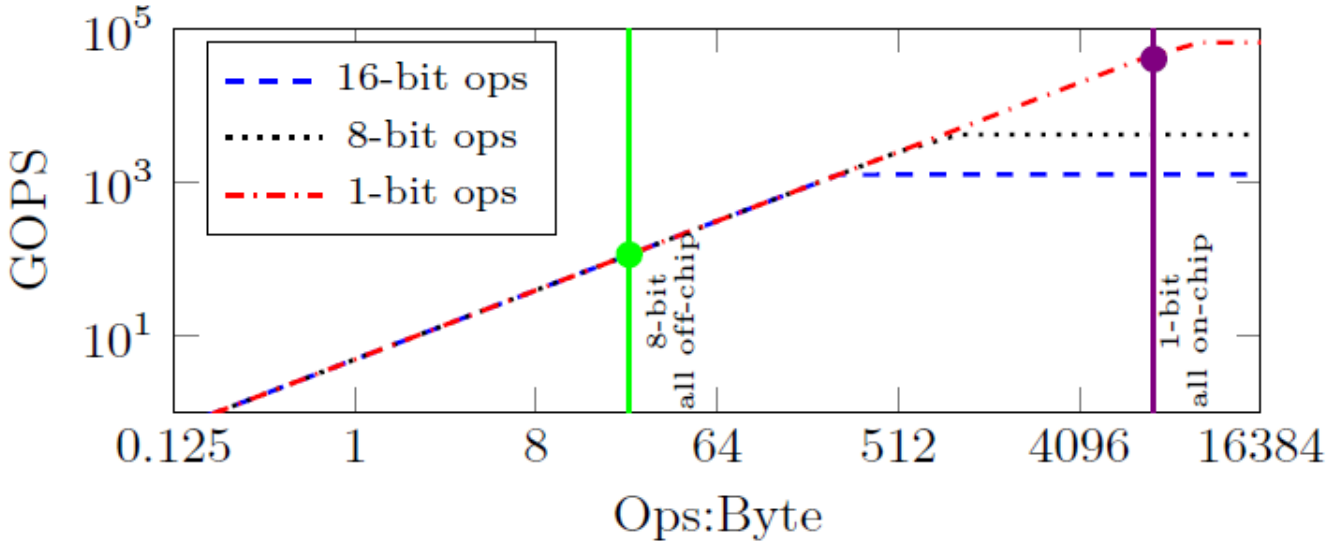



1. High AI application.
   - No cache reuse again
   - Compute bound but not using vectorization/FMA/both VPUs
2. Implement vectorization
   - Since we are not touching memory, the AI in roofline does not change
   - We are fully utilizing VPUs → FLOPS increases
3. Implement FMA use

[1] S. Williams et al. *CACM* (2009), crd.lbl.gov/departments/computer-science/PAR/research/roofline

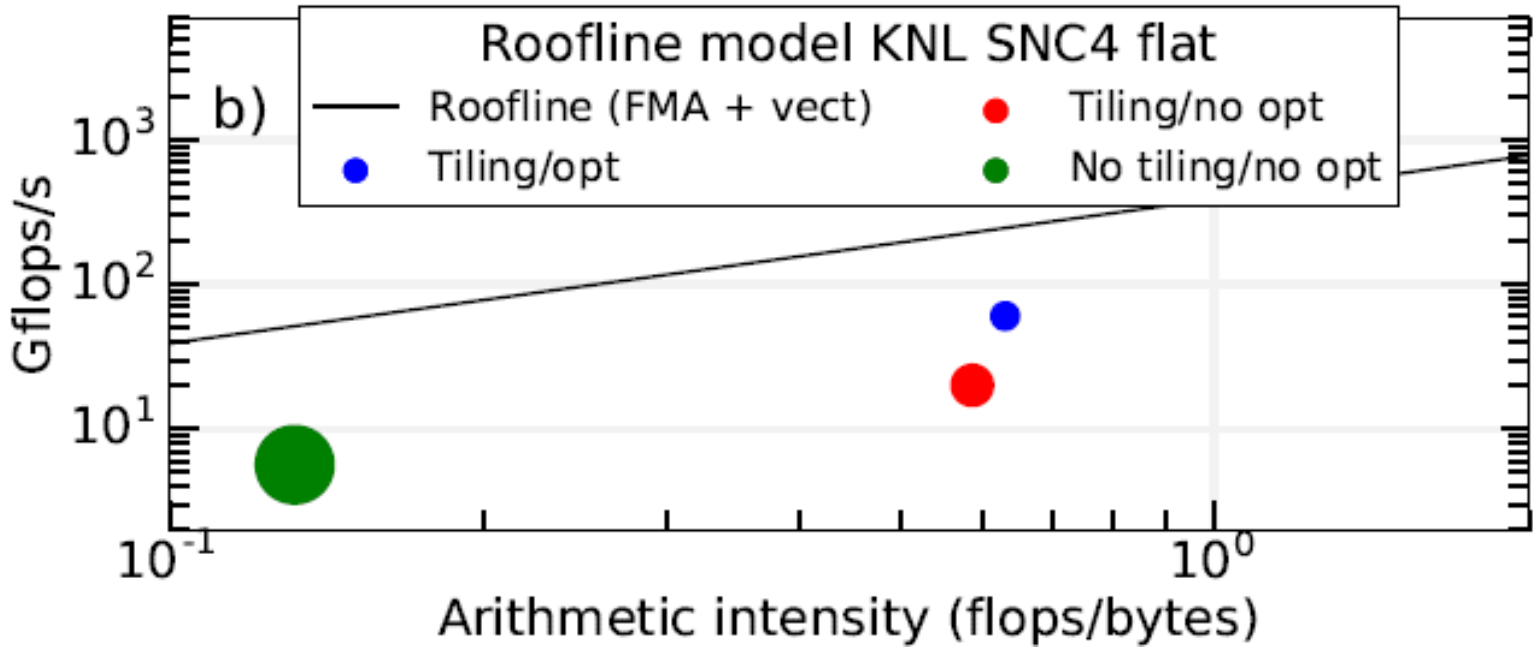
# Impact of lowering the precision

- For AlexNet, compute-bound performance for binary operations is
  - 16 times higher than 8b fixed-point and
  - 53 times higher than 16b fixed-point operations.



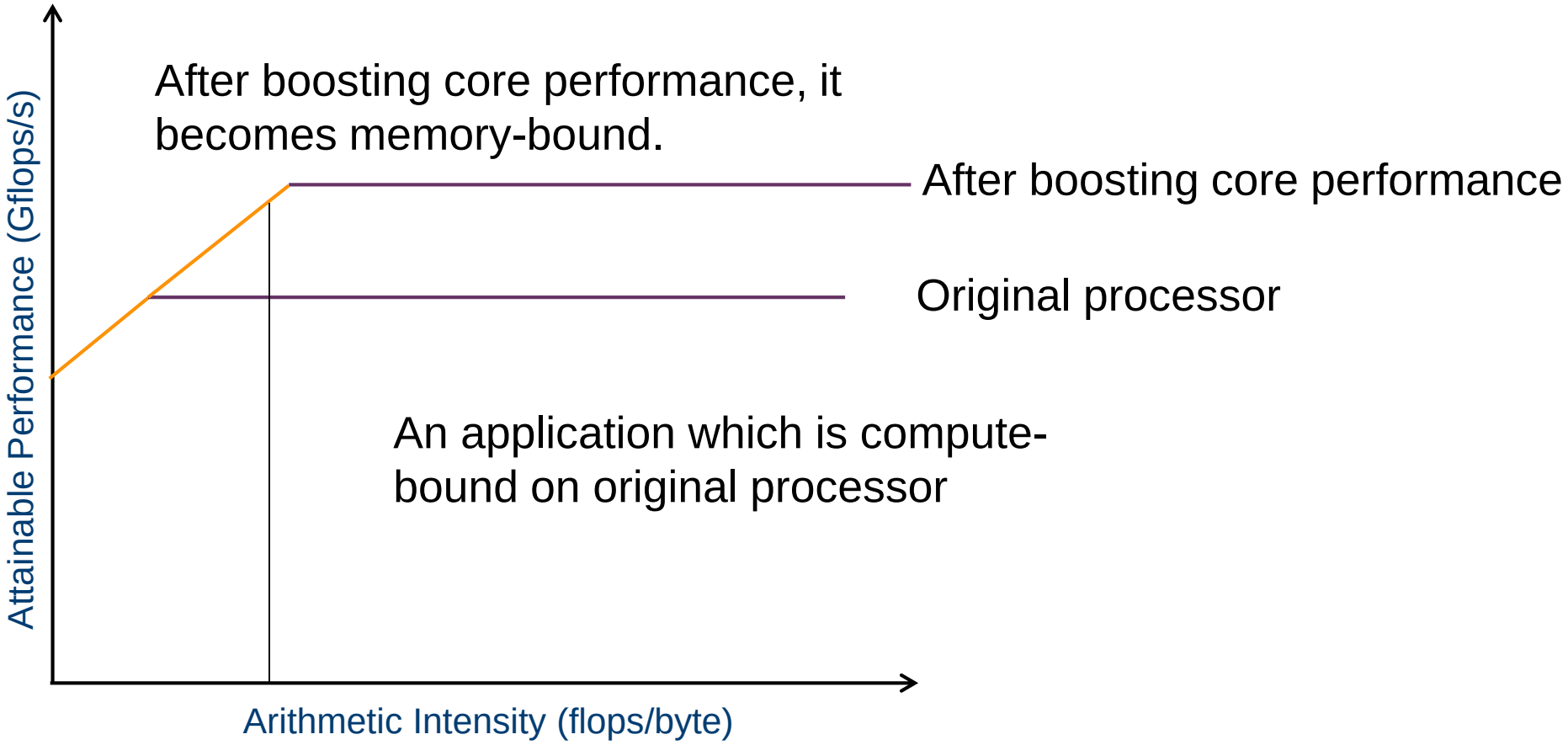

Roofline model of an FPGA. Vertical lines of AI are for AlexNet

# Impact of tiling and code-optimizations




Roofline model of Knights Landing (KNL) Phi
(Dots are for PICSAR application)

# Achieving balance between compute and memory



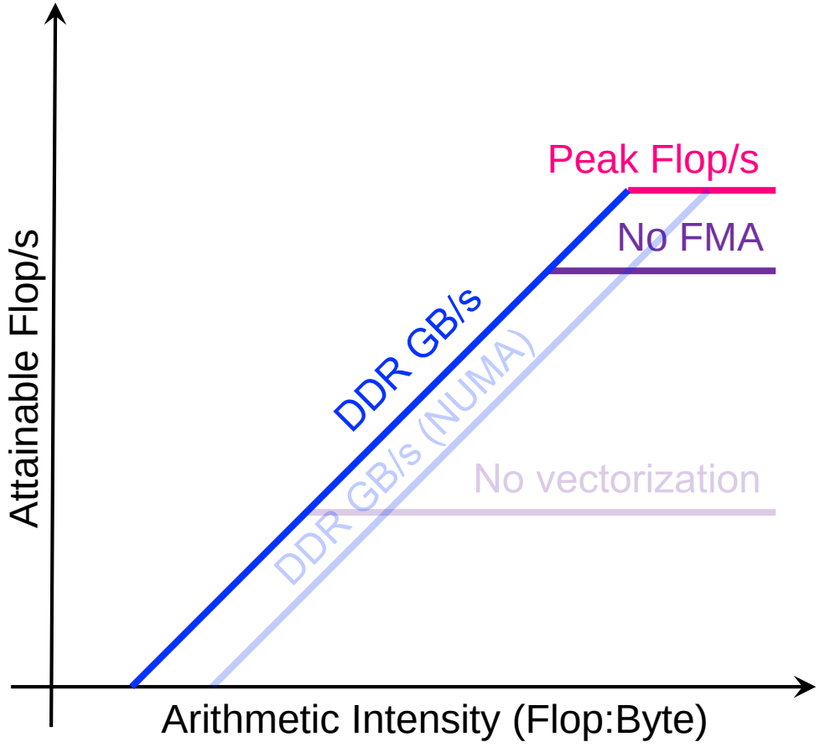

**Lesson:** optimizing only one of memory or compute is not good. We need to optimize both and balance them.

## Some data from real processors

- For GPUs, op to byte ratios have risen from 18 with the K80 to 139 for the V100.
- Google's TPUv2 has an op to byte ratio of 300.
- For AWS Inferentia, op to byte ratio is expected to be almost 500.

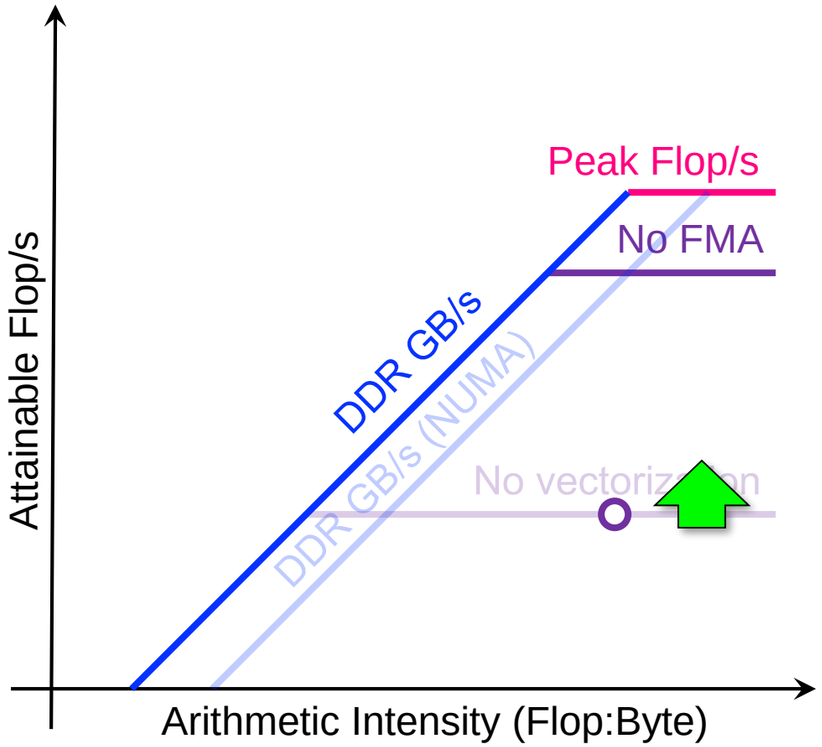
# Driving Performance Optimization

- Broadly speaking, there are three approaches to improving performance:



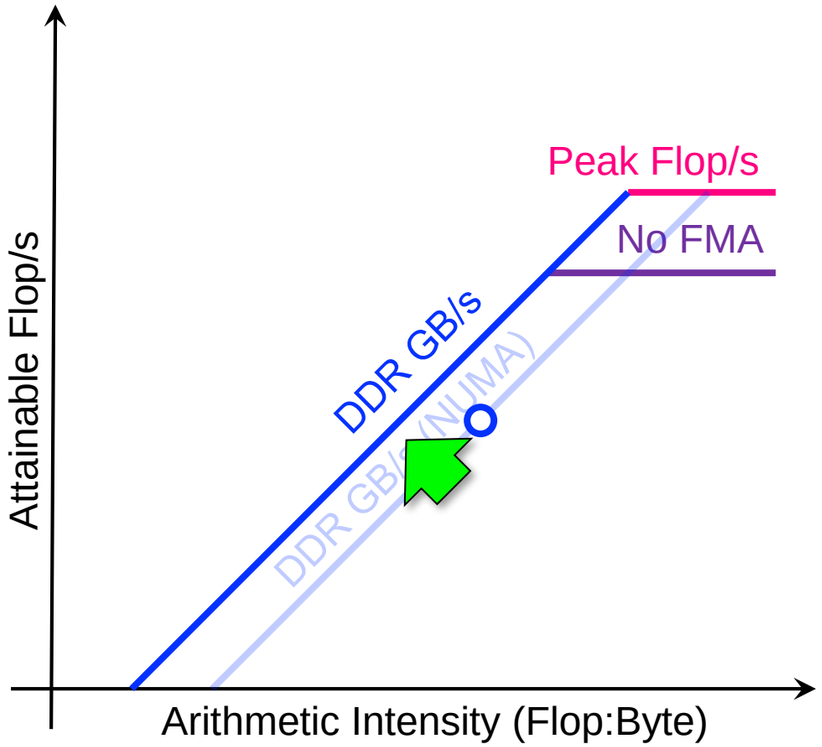
# Driving Performance Optimization

- Broadly speaking, there are three approaches to improving performance:
- **Maximize in-core performance (e.g. get compiler to vectorize)**



# Driving Performance Optimization

- Broadly speaking, there are three approaches to improving performance:
- Maximize in-core performance (e.g. get compiler to vectorize)
- **Maximize memory bandwidth (e.g. NUMA-aware, unit-stride)**



# Driving Performance Optimization

- Broadly speaking, there are three approaches to improving performance:
- Maximize in-core performance (e.g. get compiler to vectorize)
- Maximize memory bandwidth (e.g. NUMA-aware, unit stride)
- **Minimize data movement (e.g. cache blocking)**