CMOR 421/521:

The roofline performance model

M	W	F	
			1
			2
			3
			4
			5
			6
			7
			8
			9
			10
			11
			12
			13
			14
			15

How to measure performance?

- Computers vary in performance and evolve over time, so raw timings aren’t consistent. Would like a metric which is less sensitive to the choice of architecture, algorithm, etc.
- A simple idea: percentage of peak performance
  - Most programs run at <10% of peak, so there’s usually plenty of room to optimize.

What is “peak” performance?

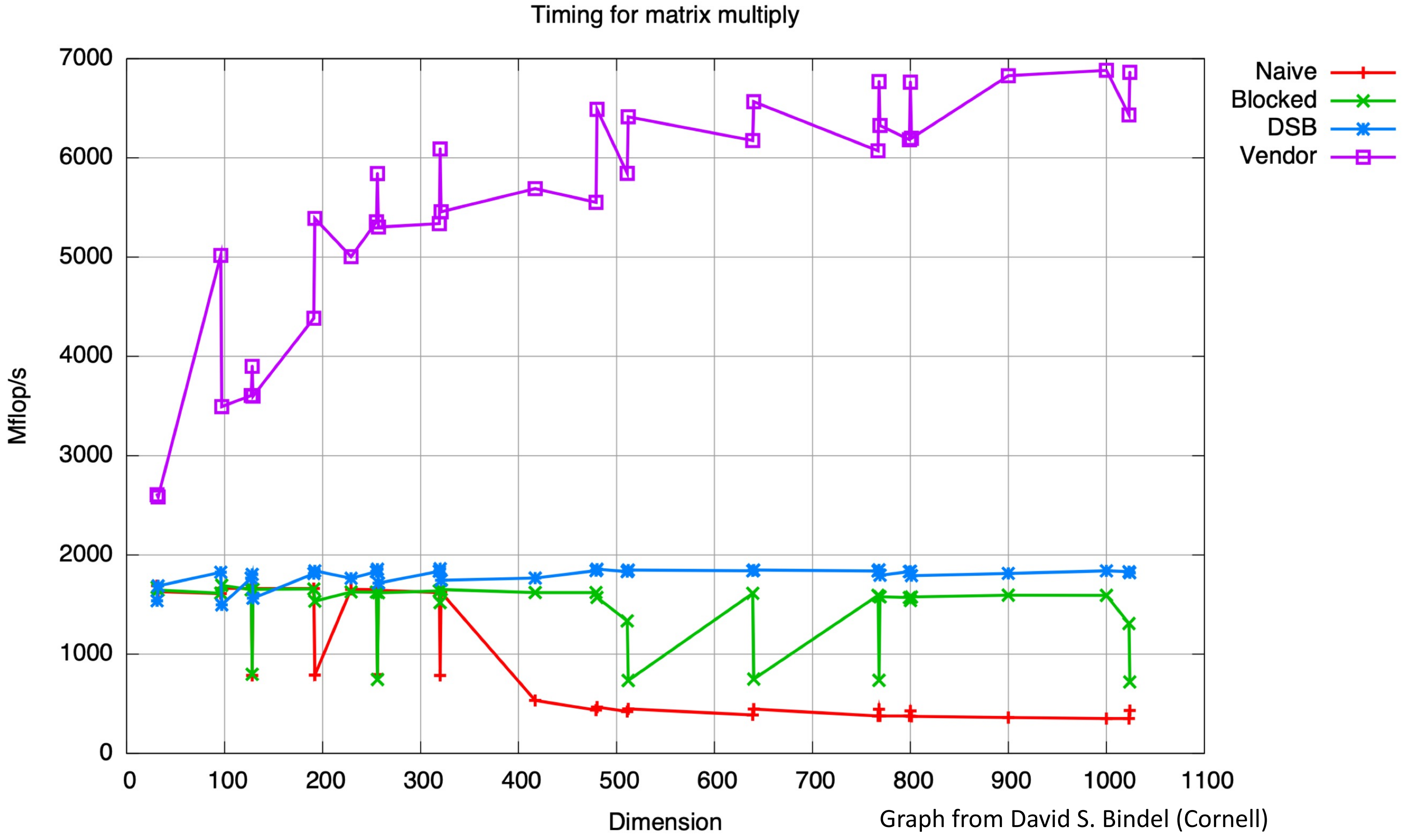
- Peak GFLOPS for a single core = (CPU speed in GHz) * (CPU instructions per cycle).
- Can also multiply by (number of CPU cores) * (number of CPUs per node).

Theoretical vs practical peak

- *Very* hard to achieve actual peak performance
- Getting close to peak performance typically requires hardware- and architecture-specific tools

Theoretical vs practical peak

Vendor implementations are often significantly faster than hardware-agnostic optimized implementations.



Graph from David S. Bindel (Cornell)

Practical considerations

- Should make sense across different architectures (single and multi core), programming models, algorithms (and parts of algorithms).
- Performance model should help guide "When should you quit optimizing?"
  - Some algorithms just don't have enough operations to achieve peak performance (e.g., matrix transposition, streaming operations)

Roofline model

- Introduced in Sam Williams’ 2008 PhD thesis
- Idea: applications typically limited by either peak computational performance or memory bandwidth
  - Matvec: bandwidth bound
  - Matmul: compute bound (if implemented efficiently)

Roofline model ingredients

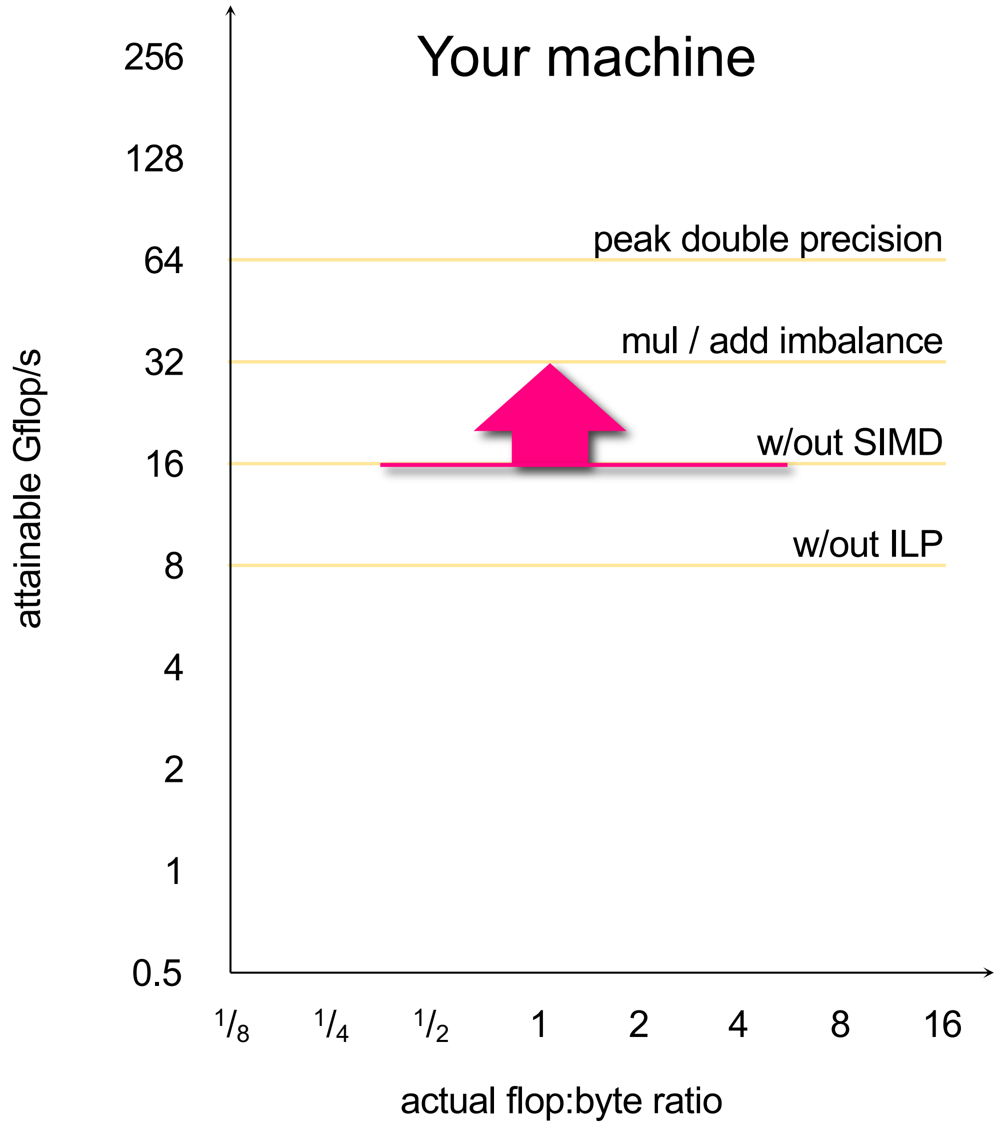
- Requires two measures of machine performance, one measure of algorithm performance
- Machine performance:
  - Arithmetic performance (flops/second)
  - Memory bandwidth (bytes/second)
- Algorithmic performance:
  - Computational intensity (flops/byte)

What is the arithmetic “roof”?

- Computer processors execute commands in a synchronized fashion; the speed of execution is the “clock rate” in GHz (event or cycle per second).
- Computers execute at most some maximum number of instructions per clock cycle
  - Multiple instructions per cycle through Instruction-level Parallelism (ILP) or Single Instruction Multiple Data (SIMD) parallelism.
- Can estimate single processor peak performance by
  (Clock rate) x (max instructions per cycle)

What is the arithmetic “roof”?

- Roof idea: different levels of attainable arithmetic performance
- Depends on lots of factors, e.g.,
  - hardware-specific compiler options
  - number of multiplies and additions (peak arithmetic performance usually assumes FMAs)



Arithmetic performance

- Flops per second is a good indication of performance for highly optimized matrix multiplication
- Not so good of an indication of performance for matrix-vector multiplication

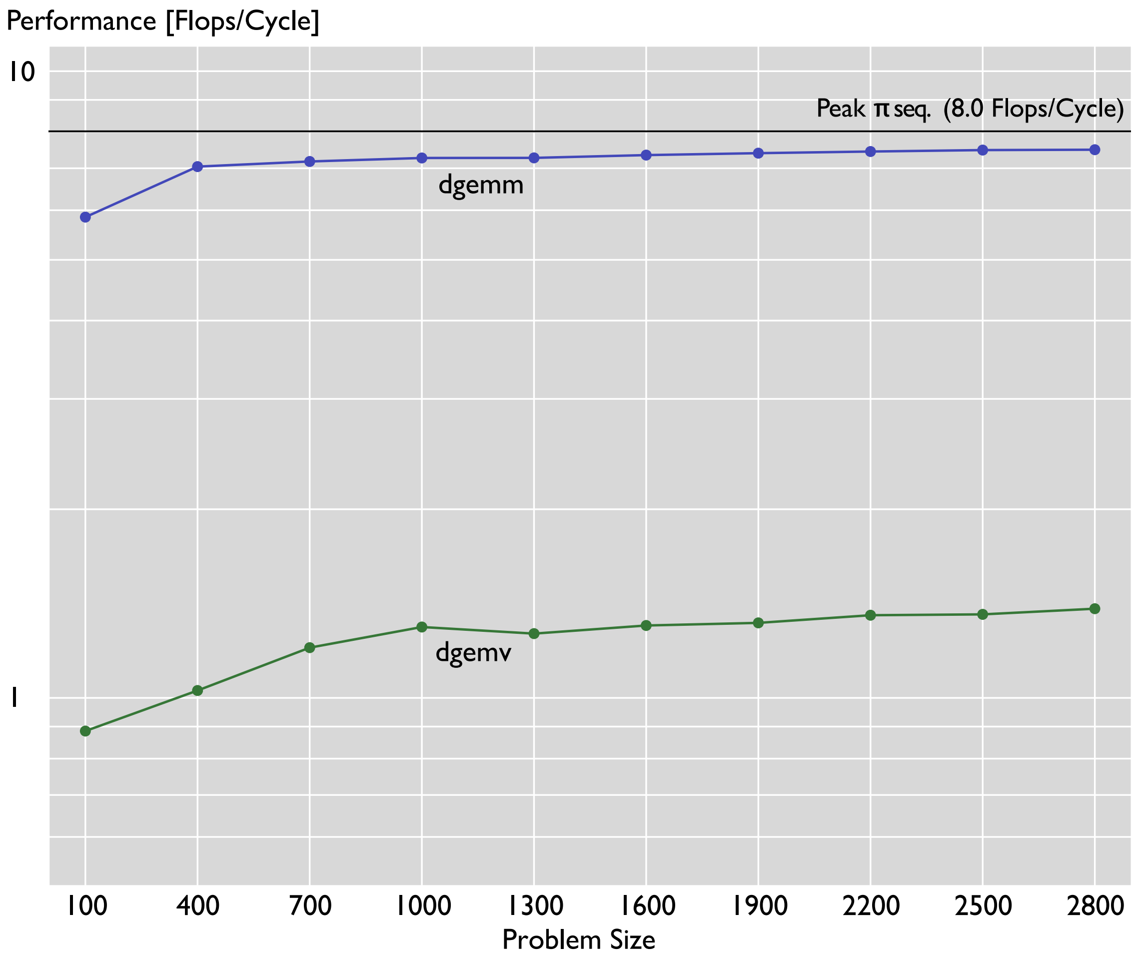


Image from G. Ofenbeck, R. Steinman, V. Caparrós Cabezas, D. Spampinato, M. Püschel

How to estimate matrix-vector performance?

- Recall that for a typical implementation, computational intensity (CI) <= 2 flops/word
  - Single precision: one word = 4 bytes, ½ flops/byte
  - Double precision: one word = 8 bytes, ¼ flops/byte

- Assume **run time ~ data movement**, which we *bound from below* by the size of input and output.
  - Estimating memory movement by this lower bound + CI provides an upper bound on possible performance.

Machine "balance"

- Defined as

  (Peak FLOPS/second) / (Peak bytes/second)
  - On modern machines, ~5-10 FLOPS per byte (and growing)
- CI = FLOPS performed / data moved, can compare to machine balance.

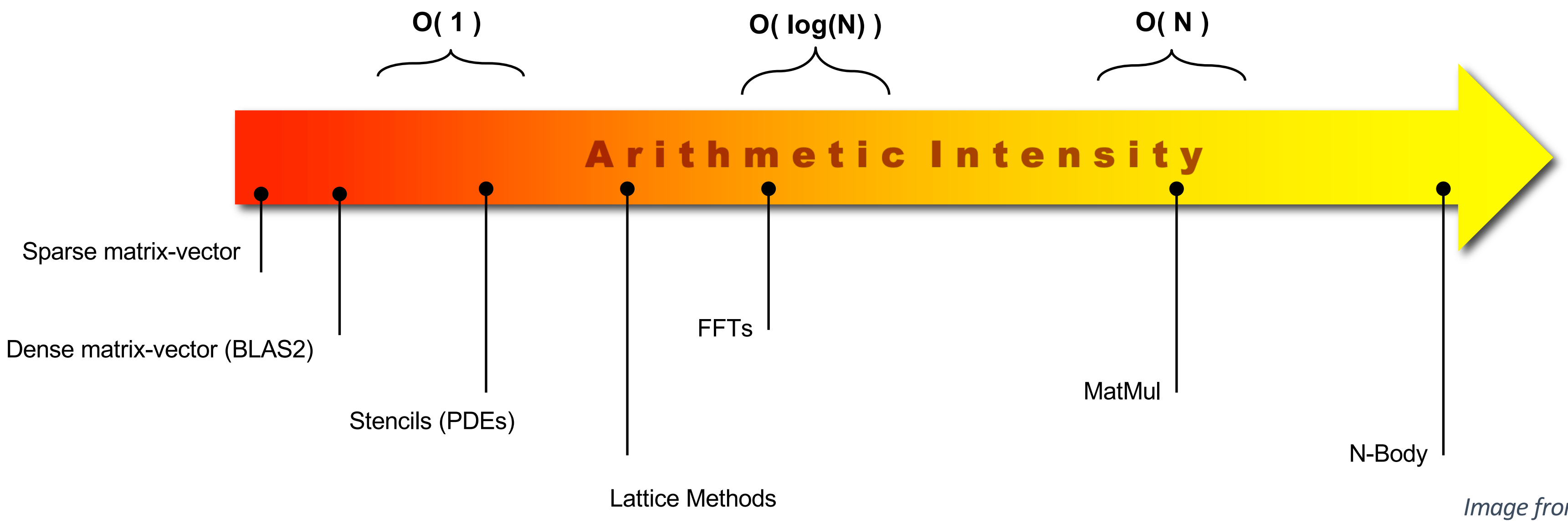


Image from Demmel

Incorporating memory into roofline

- Assumes data starts in RAM and idealized cache
- Roofline estimate: computational runtime bounded by

```
runtime = max((# flops) / (peak flops/sec),
              (# bytes) / (peak bytes/sec))
```

Incorporating memory into roofline

- Assumes data starts in RAM and idealized cache
- Roofline estimate: computational runtime bounded by

```
runtime = max((# flops) / (peak flops/sec),
              (# bytes) / (peak bytes/sec))
```

- This implies that
  (# flops) / runtime =
  1 / max(1 / (peak flops/sec),
          (# flops) / ((# bytes) / (peak bytes/sec))).

Incorporating memory into roofline

- (# flops) / runtime =
  1 / max(1 / (peak flops/sec),
          (# bytes) / ((# flops) / (peak bytes/sec))).

- Observe that 1 / max(1/a, 1/b) = min(a, b)

Incorporating memory into roofline

- (# flops) / runtime =
  1 / max(1 / (peak flops/sec),
  (# bytes) / ((# flops) / (peak bytes/sec))).

- Observe that 1 / max(1/a, 1/b) = min(a, b)

- (# flops) / runtime =
  min(peak flops/sec,
  (# flops) / (# bytes) * (1/(peak bytes/sec)))

This is computational intensity (CI)

Plotting the roofline

- (# flops) / runtime =
  min(peak flops/sec, CI / (peak bytes/sec))

Image from Wikipedia

Plotting the roofline

- (# flops) / runtime =
  min(peak flops/sec, CI / (peak bytes/sec))

- Peak flops/sec >> peak bytes/sec (memory movement).
- As CI increases, we transition linearly from memory bound to compute bound behavior.

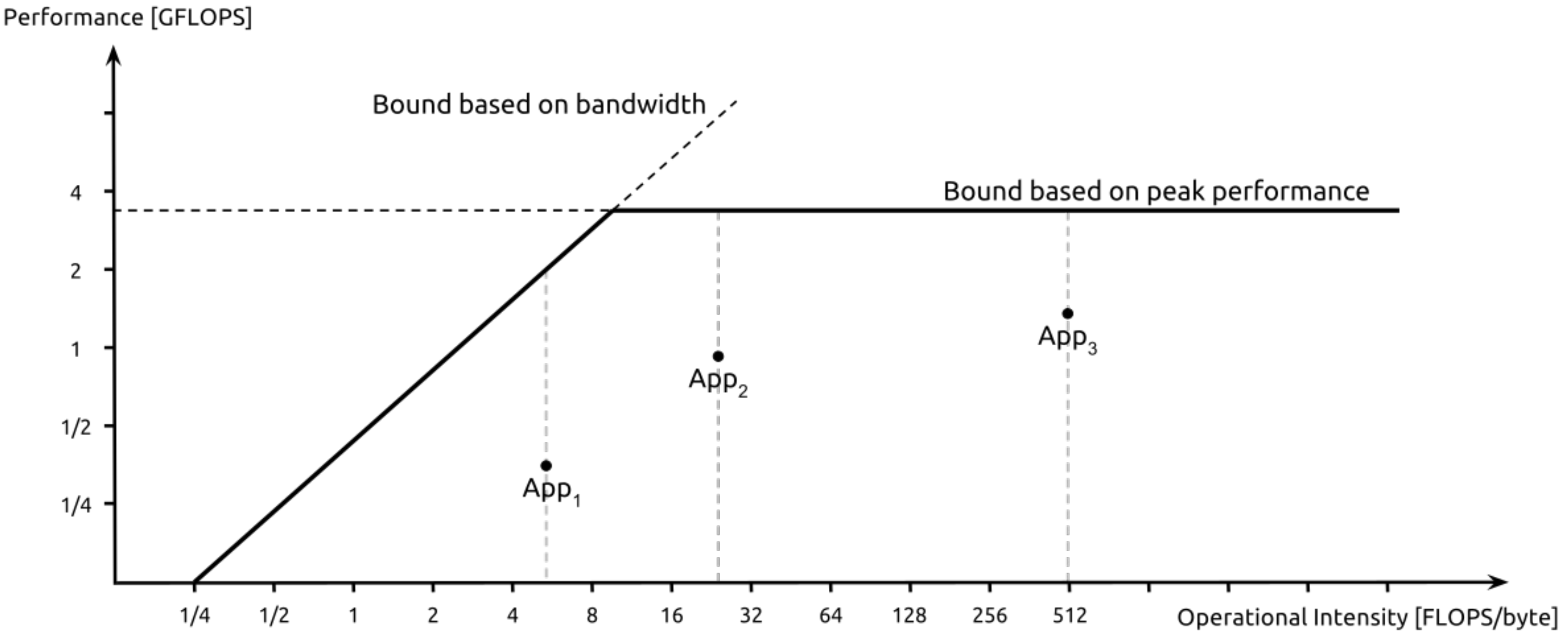
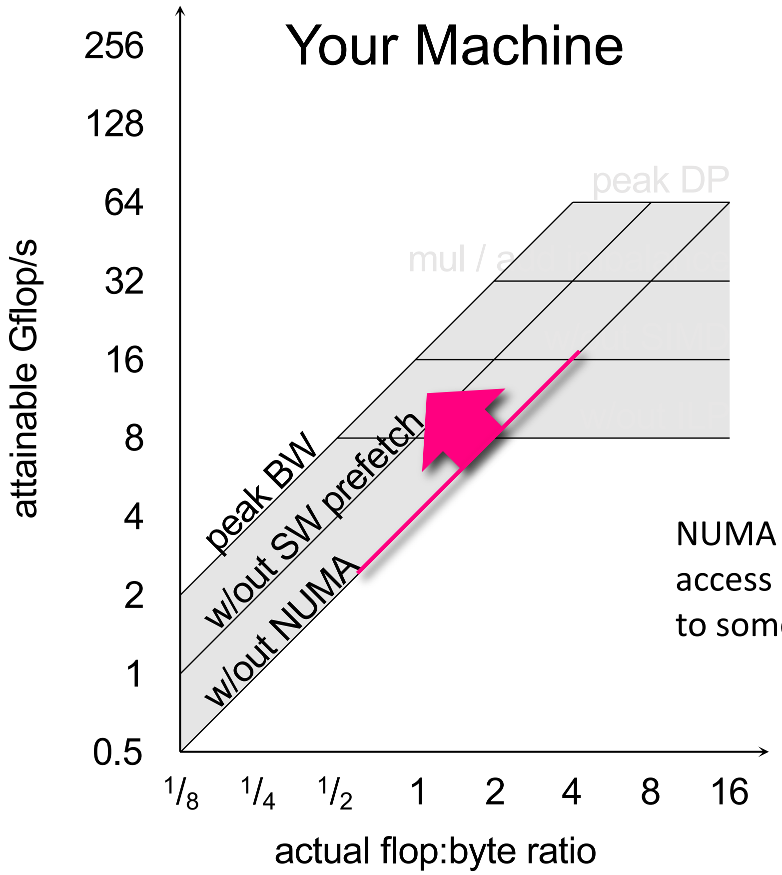


Image from Wikipedia

Different memory roofs

- Peak bandwidth is also only attainable after several layers of optimizations
  - Software/manual prefetching
  - Exploiting memory access priority
  - Different roofs for levels of memory



NUMA = non-uniform memory access (some memory closer to some cores)

Image from Demmel

Roofline summary

- Roofline tells you where you might be able to find more performance.
- Can try to increase CI (algorithmic changes)
- Can try to get closer to the roof (hardware-aware optimizations)

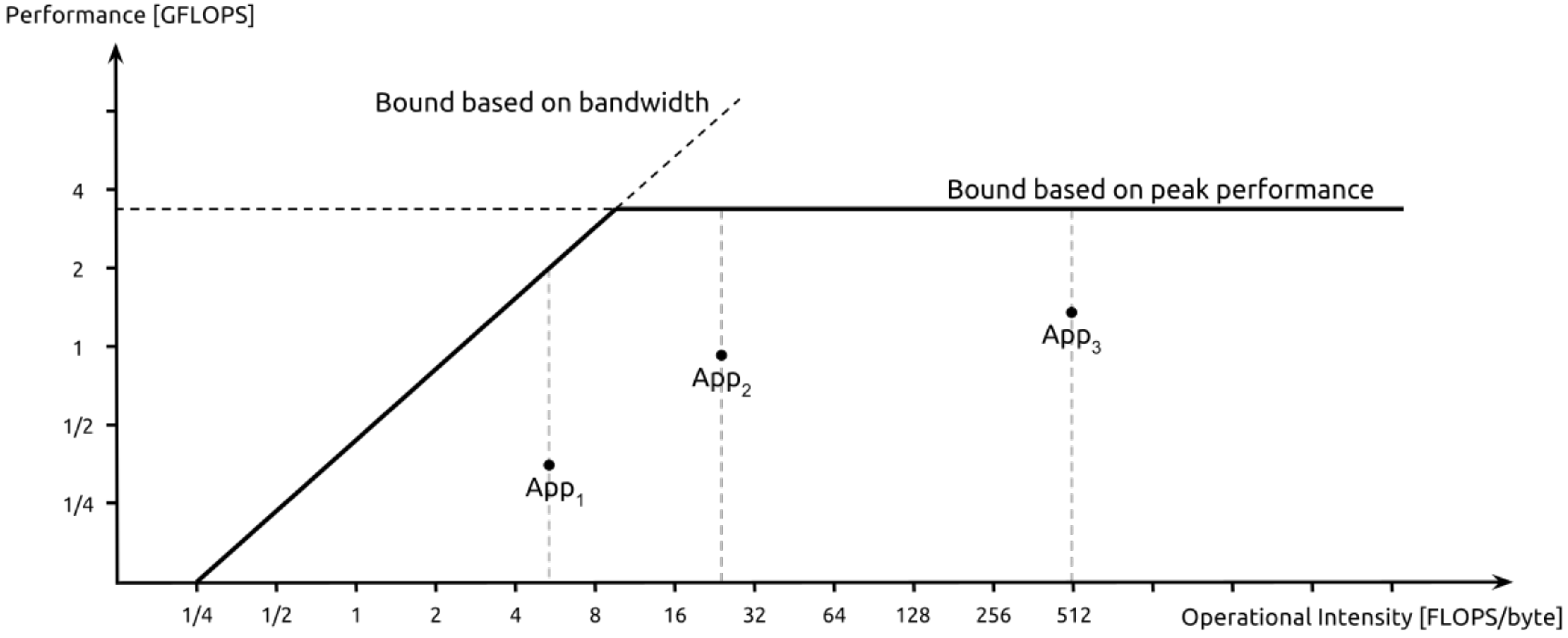


Image from Wikipedia

Roofline example

- GPU implementation of some numerical method; similar to (small matrix A) * (large matrix B)