LECTURE 6

ROOFLINE MODEL

Miquel Pericàs

EDA284/DIT361 - 2019/2020

What's cooking

1. **Lectures**
   - Today: **Intro to Roofline Model**
   - Next week: Core Multithreading and Lab Intro (Tue 9h), Chip Multiprocessors (Wed 8h) + Lab session on Friday
2. **Practice session (today ~10h)**
   - Problems 1.2 (a-e), 3.27, 5.2, 2019 Re-exam Problem 3,
3. **EDA284 project**
   - Finalized teams and selected topics are now in Canvas
   - https://chalmers.instructure.com/courses/8752/files/folder/Project#
   - Remember to contact your partners ASAP and make sure that they are available for the project!
   - Detailed instructions on the submission will be published soon

OUTLINE (Lecture 6)

- Performance Modeling
- DRAM Roofline
- Hierarchical Roofline Model

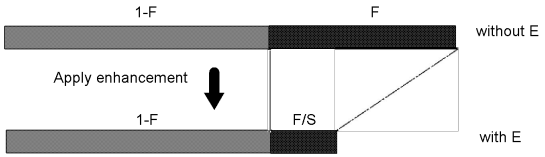
Acknowledgements

This presentation is based on materials by Samuel Williams, Computational Research Division, Lawrence Berkeley National Lab, in particular the following video lecture: https://www.youtube.com/watch?v=8h3f3E-Oq5A

Why Use Performance Models or Tools?

- Identify performance bottlenecks
- Motivate software optimizations

Amdahl's law



- **Determine when we're done optimizing**
  - Assess performance relative to machine capabilities
  - Motivate need for algorithmic changes

Peak (Fl)ops, Peak BW

- Predict performance on future machines / architectures
  - Sets realistic expectations on performance for future procurements
  - Used for HW/SW Co-Design to ensure future architectures are well-suited for the computational needs of today's applications.

Performance Models / Simulators

- Historically, many performance models and simulators tracked latencies to predict performance (i.e. counting cycles)
  - e.g. Gem5 (EDA284 Labs)

- The last two decades saw a number of latency-hiding techniques…
  - Out-of-order execution (hardware discovers parallelism to hide latency)
  - HW stream prefetching (hardware speculatively loads data)
  - Massive thread parallelism (independent threads satisfy the latency-bandwidth product)

- Effective latency hiding has resulted in a shift from a latency-limited computing regime to a **throughput-limited computing regime**
  - **Depending on the use case, latency or throughput are main goals**
  - Latency: Execution Time, Time to solution
  - Throughput: Requests per second, Multiprogrammed workloads (cloud), etc

Roofline Model

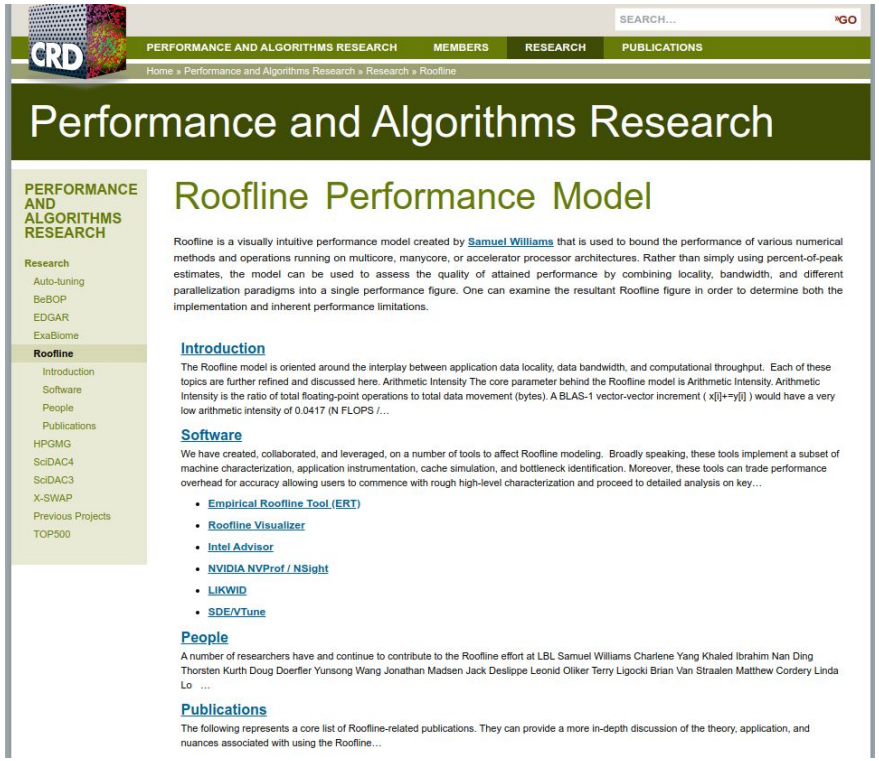
- The **Roofline Model** is a throughput-oriented performance model…
  - Tracks rates not times
  - Augmented with Little's Law

    (concurrency = latency*bandwidth)
  - Independent of ISA and architecture

    (applies to CPUs, GPUs, Google TPUs[1], etc…)
- Components:
  - Machine Characterization

    (realistic performance potential of the system)
  - Application Characterization

    (compute and memory needs)

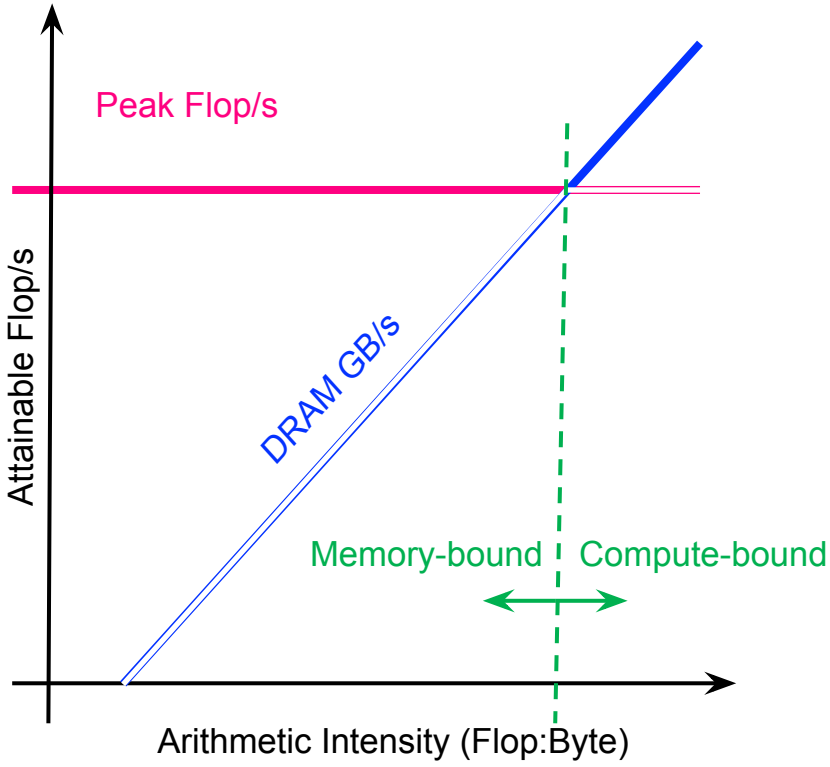


https://crd.lbl.gov/departments/computer-science/PAR/research/roofline

[1]Jouppi et al, "In-Datacenter Performance Analysis of a Tensor Processing Unit", ISCA, 2017.

Simplified (DRAM) Roofline

- One could hope to always attain peak performance (Flop/s)
- However, finite locality (reuse) and bandwidth limit performance.
- Consider idealized processor/caches
- Plot the **performance bound** using Arithmetic Intensity (AI) as the x-axis…
  - AI = Flops / Bytes presented to DRAM
  - **Attainable Flop/s = min(peak Flop/s, AI * peak GB/s)**
  - **Log-log scale** makes it easy to represent whole range
  - Kernels with AI less than machine balance are ultimately DRAM bound (we'll refine this later…)



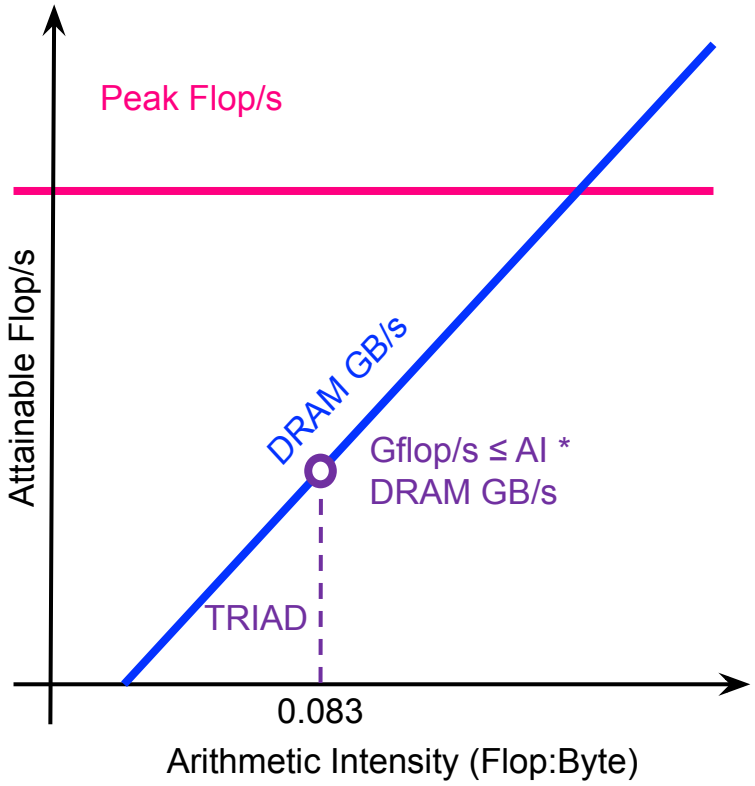
Roofline Example #1

- Typical machine balance is 5-10 flops per byte…
  - 40-80 flops per double to exploit compute capability
  - Artifact of technology and cost
  - **Unlikely to improve**
  - huge amount of reuse is needed to achieve peak flops

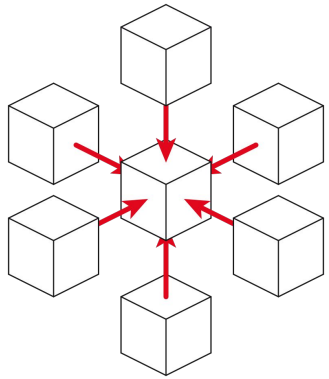
- Consider STREAM Triad…

```
#pragma omp parallel for
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**

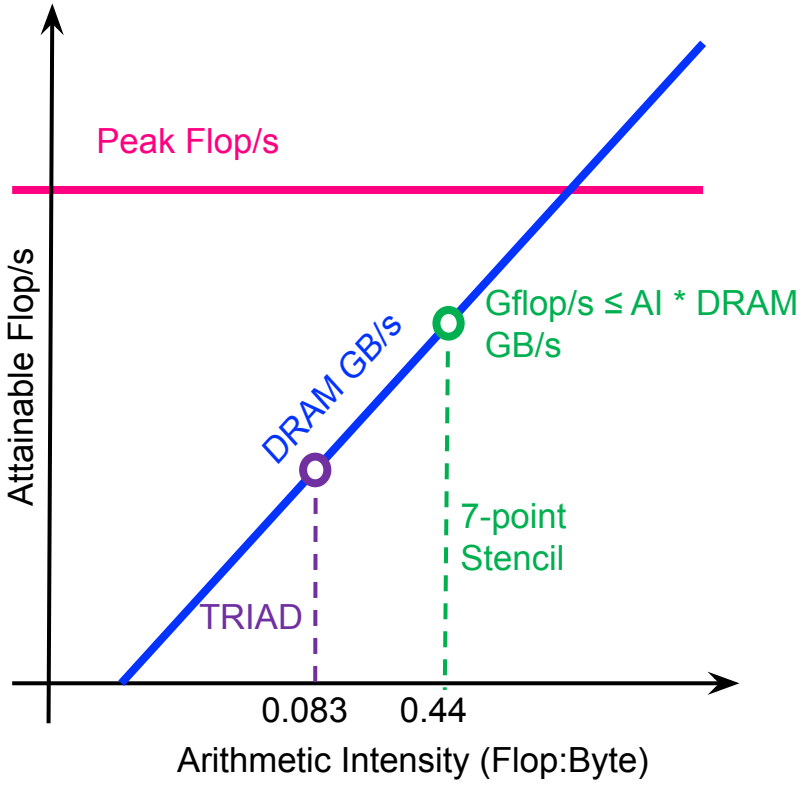


Roofline Example #2



- Conversely, 7-point (3D) stencil…
  - 7 flops
  - 8 memory references (7 reads, 1 store) per point
  - Cache can filter all but 1 read and 1 write per point
  - **AI = 0.44 flops per byte == memory bound, but 5x the flop rate**

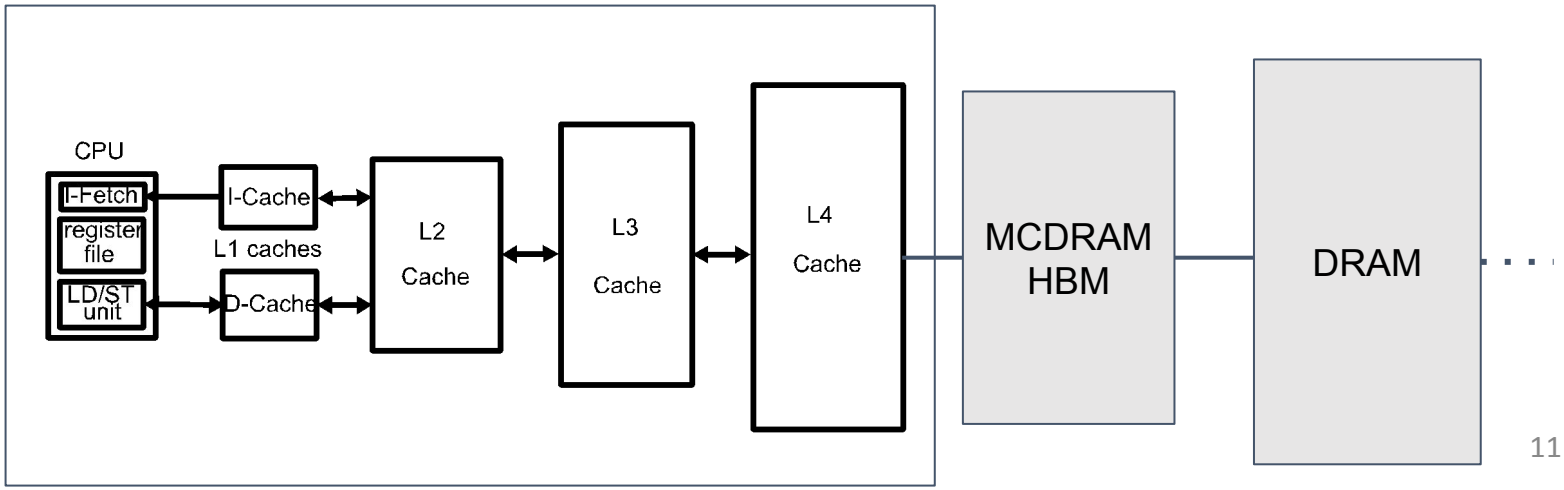
```
#pragma omp parallel for
for(k=1;k<dim+1;k++){
  for(j=1;j<dim+1;j++){
    for(i=1;i<dim+1;i++){
      int ijk = i + j*jStride + k*kStride;
      new[ijk] = -6.0*old[ijk      ]
                 + old[ijk-1       ]
                 + old[ijk+1       ]
                 + old[ijk-jStride]
                 + old[ijk+jStride]
                 + old[ijk-kStride]
                 + old[ijk+kStride];
}}}
```



Think: how much "new" data is required by the application/ kernel/loop and how many ops are performed on the data

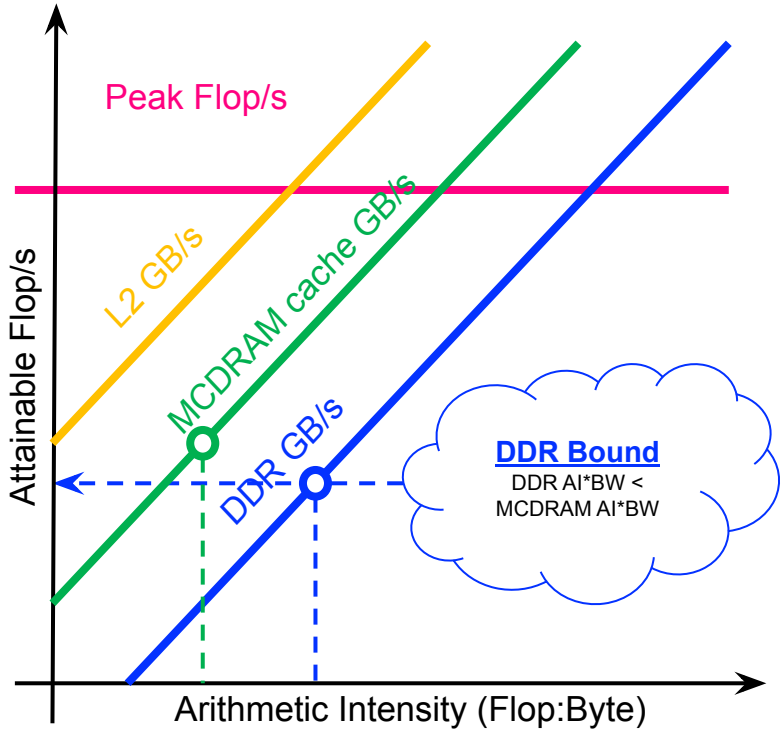
Hierarchical Roofline

- Real processors have multiple levels of memory
  - Registers
  - L1, L2, L3 cache
  - MCDRAM/HBM (KNL/GPU device memory)
  - DDR (main memory)
  - NVRAM (non-volatile memory)
- Each level loads a smaller subset of data (caches filter locality)
  - Unique data movements imply unique AI's
  - Moreover, each level will have a unique bandwidth (machine characteristic)



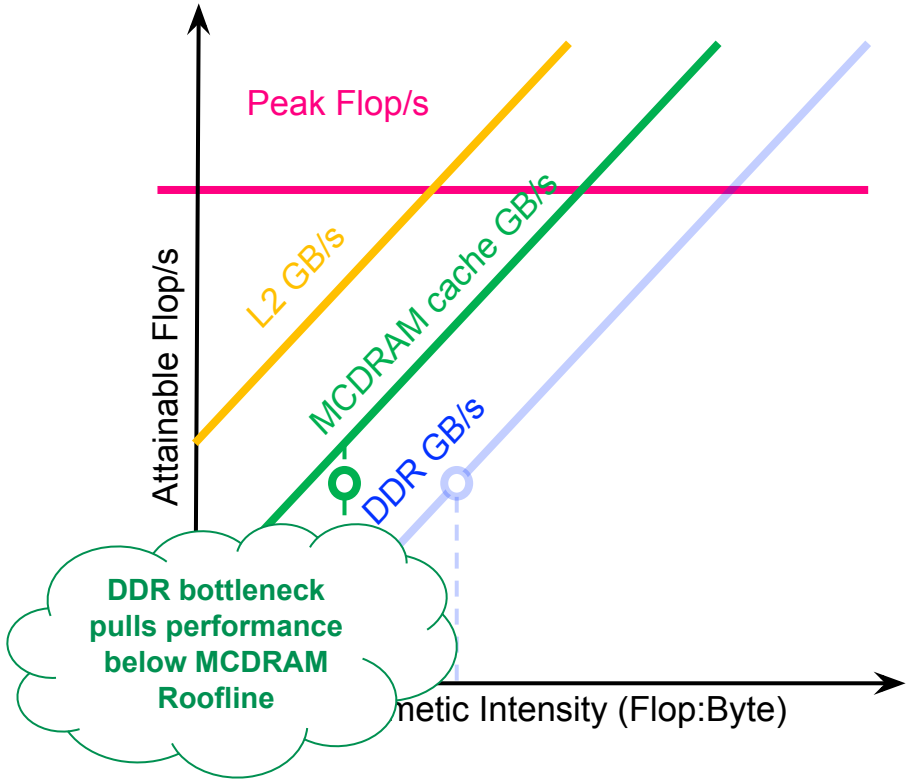
Hierarchical Roofline

- Construct superposition of the multiple Rooflines…
  - Measure a bandwidth and AI for each level of memory
  - Although a given application/kernel/loop may have multiple AI's and multiple bounds (flops, L1, L2, … DRAM)…
  - **Performance is bound by minimum**



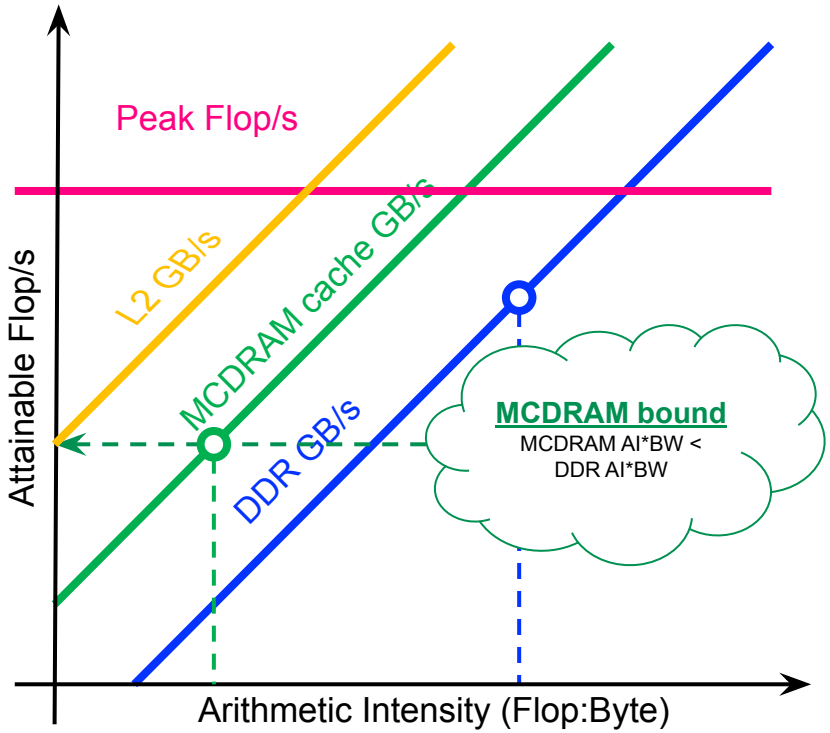
Hierarchical Roofline

- Construct superposition of the multiple Rooflines…
  - Measure a bandwidth and AI for each level of memory
  - Although a given application/kernel/loop may have multiple AI's and multiple bounds (flops, L1, L2, … DRAM)…
  - **Performance is bound by minimum**



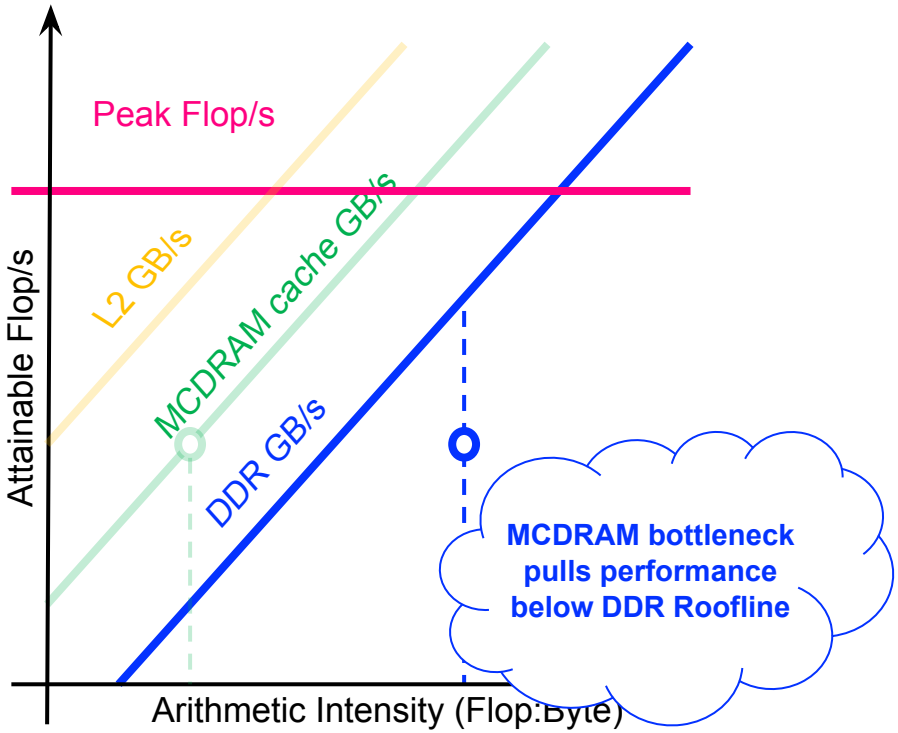
Hierarchical Roofline

- Construct superposition of the multiple Rooflines…
  - Measure a bandwidth and AI for each level of memory
  - Although a given application/kernel/loop may have multiple AI's and multiple bounds (flops, L1, L2, … DRAM)…
  - **Performance is bound by minimum**



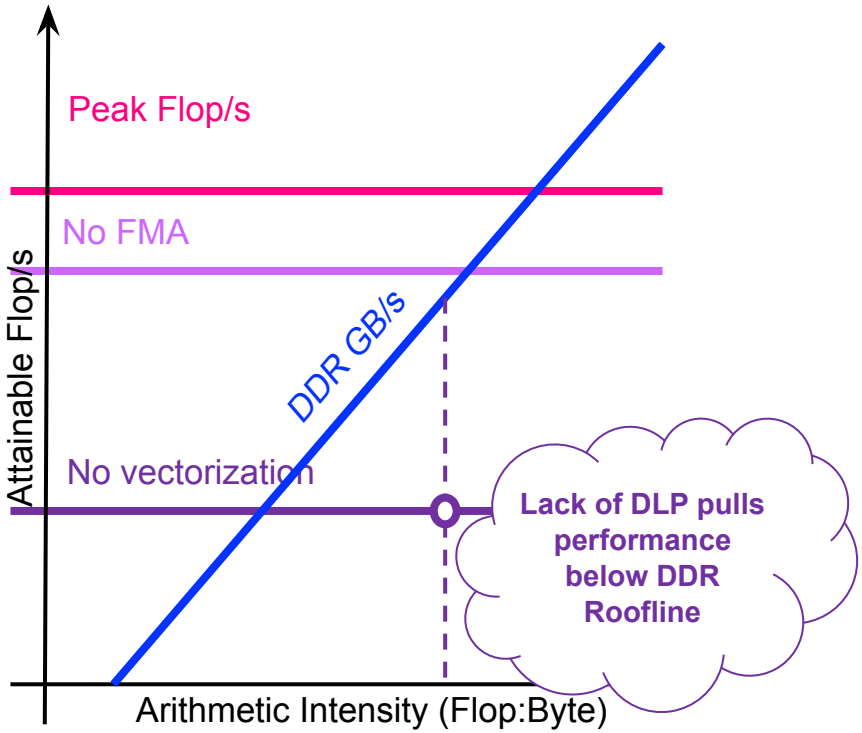
Hierarchical Roofline

- Construct superposition of the multiple Rooflines…
  - Measure a bandwidth and AI for each level of memory
  - Although a given application/kernel/loop may have multiple AI's and multiple bounds (flops, L1, L2, … DRAM)…
  - **Performance is bound by minimum**



Data, Instruction, Thread-Level Parallelism…

- We have assumed one can attain peak flops with high locality.
- In reality, this is premised on sufficient…
  - Use special instructions (e.g. fused multiply-add)
  - Vectorization (16 flops per instruction)
  - unrolling, out-of-order execution
  - OpenMP across multiple cores
- Without these:
  - Peak performance is not attainable
  - Some kernels can transition from memory-bound to compute-bound
  - DRAM bandwidth is often tied to DLP and TLP (single core can't saturate BW w/scalar code)



How to: Machine Characterization and Application Characterization (AI)?

- ***Machine Characterization:*** via microbenchmarks to measure BW at each level and max FLOPS (https://crd.lbl.gov/departments/computer-science/PAR/research/roofline/software/ert/)
- ***Application Characterization:*** measure AI via performance counters. But be careful:
  - ✘ Flop Counters can be broken/missing in production processors
  - ✘ Counting Loads and Stores doesn't capture cache reuse while counting cache misses doesn't account for prefetchers.
  - ✘ DRAM counters (Uncore PMU) might be accurate, but…
    - are privileged and thus nominally inaccessible in user mode
    - may need vendor and center approved OS/kernel changes
- **Alternatively**: use combination of software instrumentation (software Flop counters) and LLC/DRAM counters

For the EDA284 project: Think about algorithmic intensity, i.e. FLOPS, vectorization, parallelism, bytes and memory access patterns; and perform a pencil & paper roofline analysis of your scenario (algorithm) and proposed computer architecture.

There are two Major Roofline Formulations:

- Hierarchical Roofline (original Roofline w/ DRAM, L3, L2, …)…
  - **Williams, et al, "Roofline: An Insightful Visual Performance Model for Multicore Architectures", CACM, 2009**
  - Defines multiple bandwidth ceilings and multiple AI's per kernel
  - Performance bound is the minimum of flops and the memory intercepts (superposition of original, single-metric Rooflines)
- Cache-Aware Roofline
  - **Ilic et al, "Cache-aware Roofline model: Upgrading the loft", IEEE Computer Architecture Letters, 2014**
  - Defines multiple bandwidth ceilings, but uses a single AI (flop:L1 bytes)
  - As one looses cache locality (capacity, conflict, …) performance falls from one BW ceiling to a lower one at constant AI

- Why Does this matter?
  - Some tools use the Hierarchical Roofline, some use cache-aware **== Users need to understand the differences**
  - Cache-Aware Roofline model was integrated into production Intel Advisor

SUMMARY

- **Performance Modeling**
- **DRAM Roofline**
- **Hierarchical Roofline**

For more details see the youtube link on slide #4 and 2 papers in previous slide