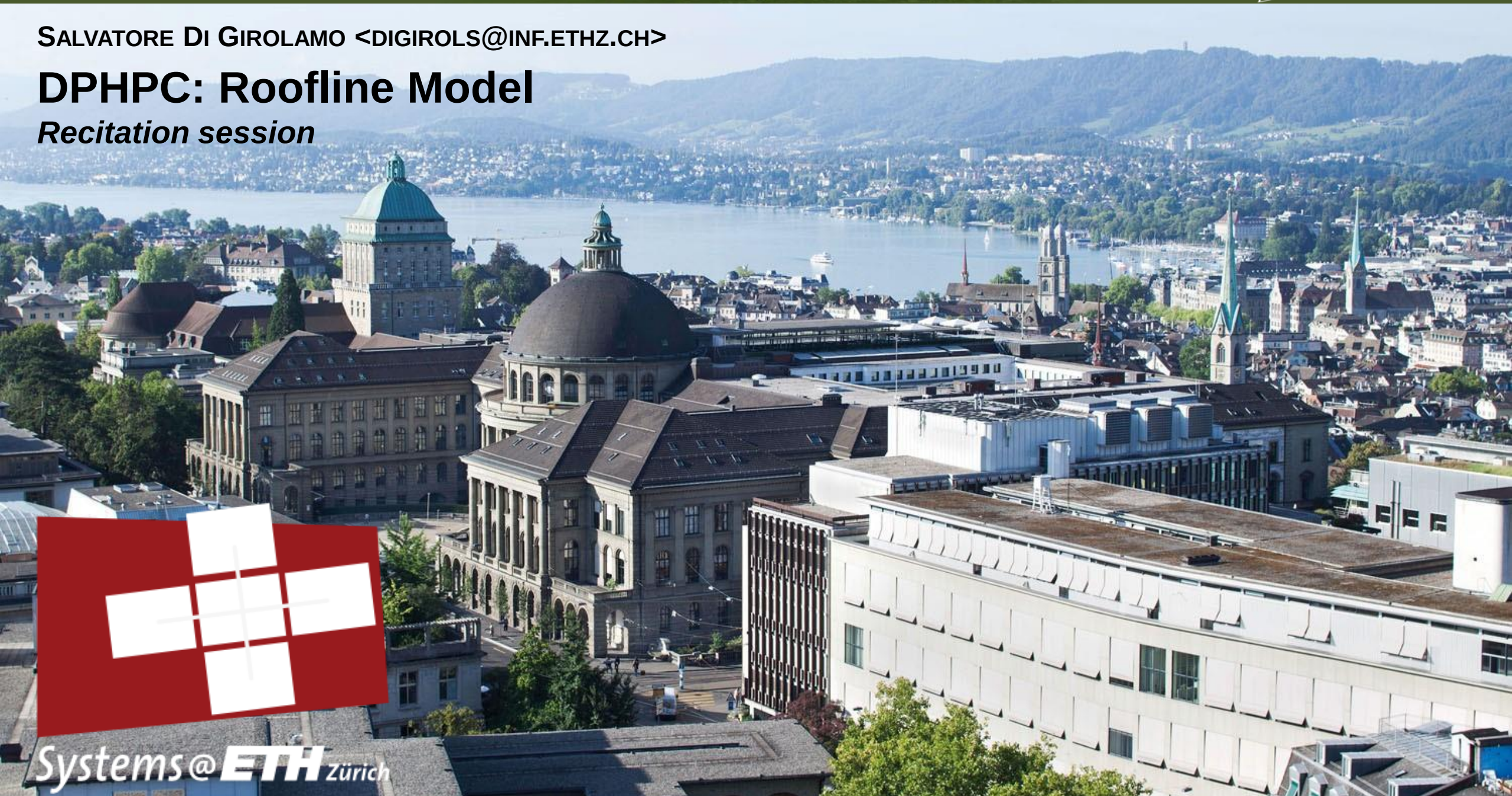
SALVATORE DI GIROLAMO <DIGIROLS@INF.ETHZ.CH>

DPHPC: Roofline Model

Recitation session



Keys to Performance

- **Computation**
- **Communication**
- **Locality**

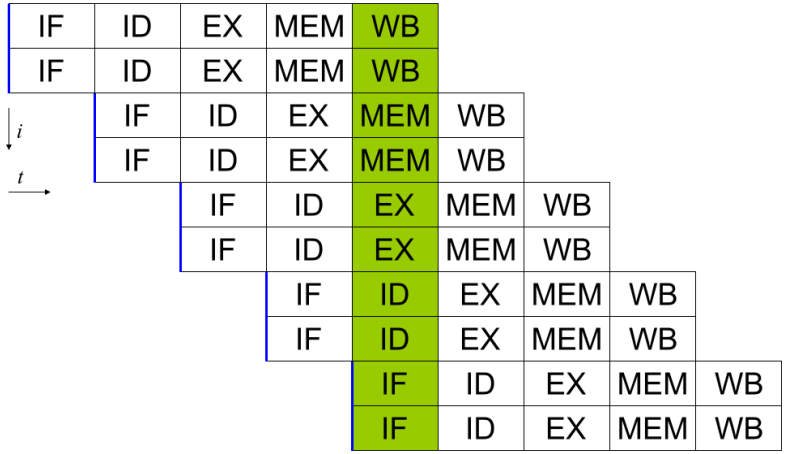
- **Each architecture has a different balance between these**
- **Each kernel has a different balance between these**

- ***Performance is a question of how well an kernel's characteristics map to an architecture's characteristics***

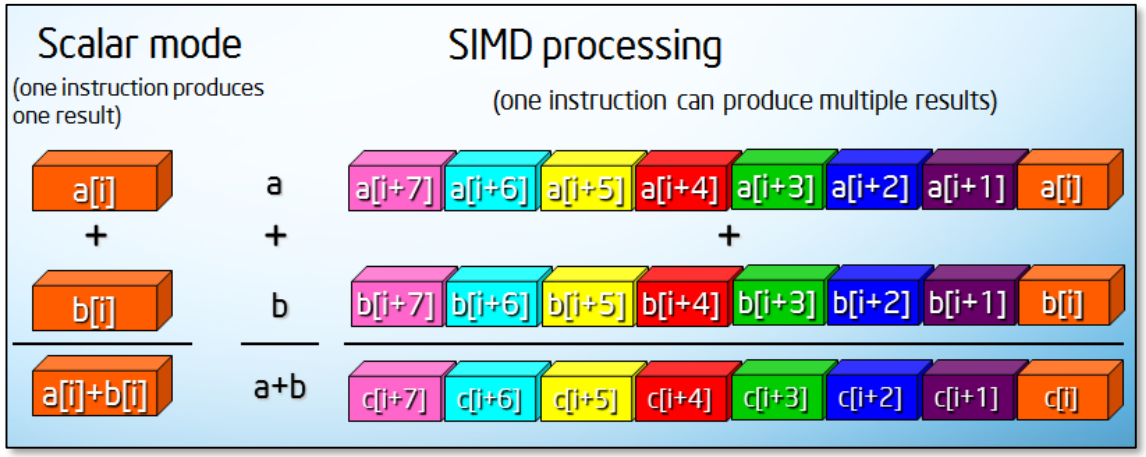
https://crd.lbl.gov/assets/pubs_presos/parlab08-roofline-talk.pdf

Computation

- **Usually, floating point performance (Gflop/s) is the metric of interest**
- **Road to peak in-core performance:**
  - ***Improve ILP and apply SIMD***



Instruction Level Parallelism (ILP)



Single Instruction Multiple Data (SIMD)

- ***Balance floating-point operation mix:*** equal number of additions and multiplications
  Hardware may have Fused Multiple-Add instructions (FMA) or equal number of adders/multipliers

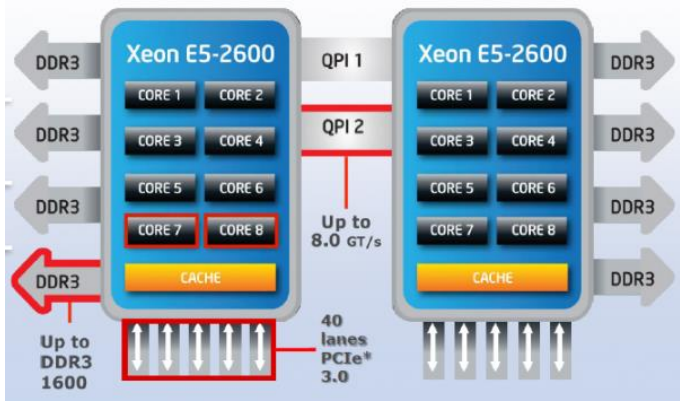
Communication

- **DRAM bandwidth (GB/s) is the metric of interest**

```
for (i=0; i<n; i++)
   for (j=0; j<n; j++)
      a[i][j] = a[i][j] + c[i][j] * d;
```

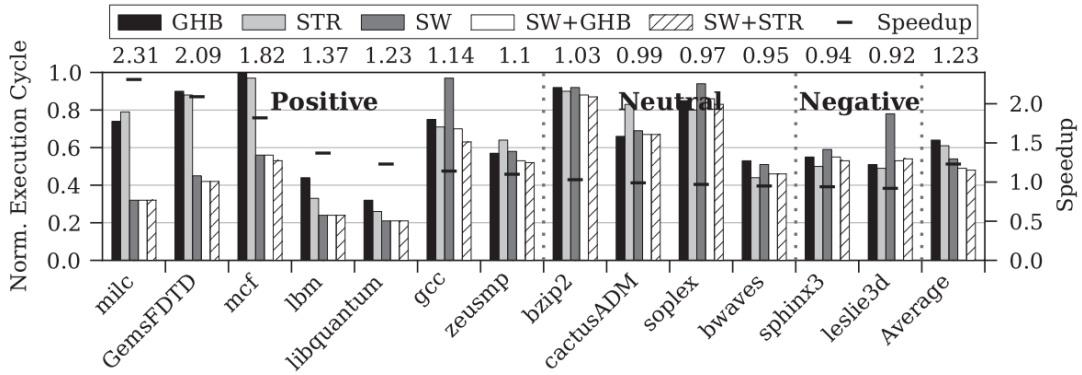
```
for (i=0; i<n; i++)
   for (j=0; j<n; j++)
      a[j][i] = a[j][i] + c[j][i] * d;
```

- **Restructure loops for unit stride accesses**
  - Engages the hardware prefetcher



- **Ensure memory affinity**
  - E.g., two multicore chips with local memory controller

- **Use software prefetching**
  - Depending on the architecture, HW prefetcher can take time (e.g., 5 loads) to start prefetching
  - SW prefetching can provide speedups for complex access patterns



Lee, Jaekyu, Hyesoon Kim, and Richard Vuduc. "When prefetching works, when it doesn't, and why." ACM Transactions on Architecture and Code Optimization (TACO) 9.1 (2012): 2.

Locality

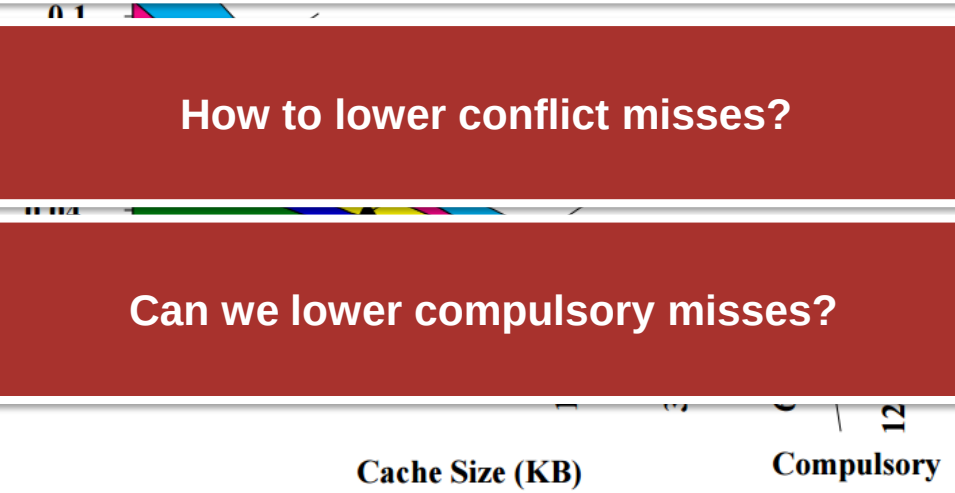
- **3Cs Model**
  - **Compulsory:** *On the first access to a block; the block must be brought into the cache; also called cold start misses, or first reference misses.*
  - **Capacity:** *Occur because blocks are being discarded from cache because cache cannot contain all blocks needed for program execution (program working set is much larger than cache capacity).*
  - **Conflict:** *In the case of set associative or direct mapped block placement strategies, conflict misses occur when several blocks are mapped to the same set or block frame; also called collision misses or interference misses.*

What is the lower bound to the number of memory operations?

How to lower capacity misses?

How to lower conflict misses?

Can we lower compulsory misses?



http://meseec.ce.rit.edu/eecc551-winter2001/551-1-30-2002.pdf

How to Improve Locality?

- **Merging Arrays**

```
/* Before: 2 sequential arrays */
int val[SIZE];
int key[SIZE];

/* After: 1 array of stuctures */
struct merge {
   int val;
   int key;
};
struct merge merged_array[SIZE];
```

- Reduce conflicts between key and val
- Improve spatial locality

- **Loop Interchange**
- **Loop Fusion**
- **Blocking or “tiling”**

How to Improve Locality?

- Merging Arrays
- Loop Interchange

```
/* Before */
for (k = 0; k < 100; k = k+1)
   for (j = 0; j < 100; j = j+1)
        for (i = 0; i < 5000; i = i+1)
              x[i][j] = 2 * x[i][j];
/* After */
for (k = 0; k < 100; k = k+1)
   for (i = 0; i < 5000; i = i+1)
        for (j = 0; j < 100; j = j+1)
              x[i][j] = 2 * x[i][j];
```

Improves spatial locality: sequential access instead of striding through memory every 100 words

- Loop Fusion
- Blocking or “tiling”

How to Improve Locality?

- Merging Arrays
- Loop Interchange
- Loop Fusion

```
/* Before */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
       a[i][j] = 1/b[i][j] * c[i][j];
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
       d[i][j] = a[i][j] + c[i][j];
/* After */
for (i = 0; i < N; i = i+1)
   for (j = 0; j < N; j = j+1)
   {   a[i][j] = 1/b[i][j] * c[i][j];
       d[i][j] = a[i][j] + c[i][j];}
```

- From two missies per access to a & c to one miss per access
- Improve spatial locality

- Blocking or “tiling”

How to Improve Locality?

- **Merging Arrays**
- **Loop Interchange**
- **Loop Fusion**
- **Blocking or “tiling”**
  - Example: matrix multiplication
  - Goal: reduce the working set

Compute/Memory Bound

- **What do we mean by "compute bound"?**
  - It has high operations intensity
- **What to we mean by "memory bound"?**
  - It has low operational intensity

- **They're not very precise definitions…**
- **Roofline model helps to clarify**
  - Plots the performance (GFlops/second) as a function of the Operational Intensity (GFlops/byte)
  - What's Operational Intensity?

Operational Intensity

- **How many Flops per byte does your code show?**
  - **Work:** W is the number of operations performed by a given program
  - **Memory Traffic:** Q is the number of bytes transferred from memory by a given program

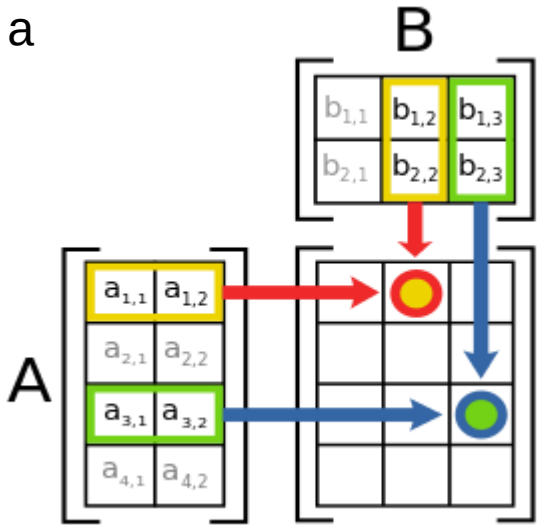
- **Can you increase it?**
  - For some kernels, OI is a function of the input size
    e.g., dense matrix multiplication
  - What else?
    Improve locality

  - **Example:** matrix multiplication (3 nested loops)

$$W(n) = \sim n^3$$
$$Q(n) = n^2$$

$$I(n) = \frac{W(n)}{Q(n)} = \sim n$$

Are we making some assumptions here?

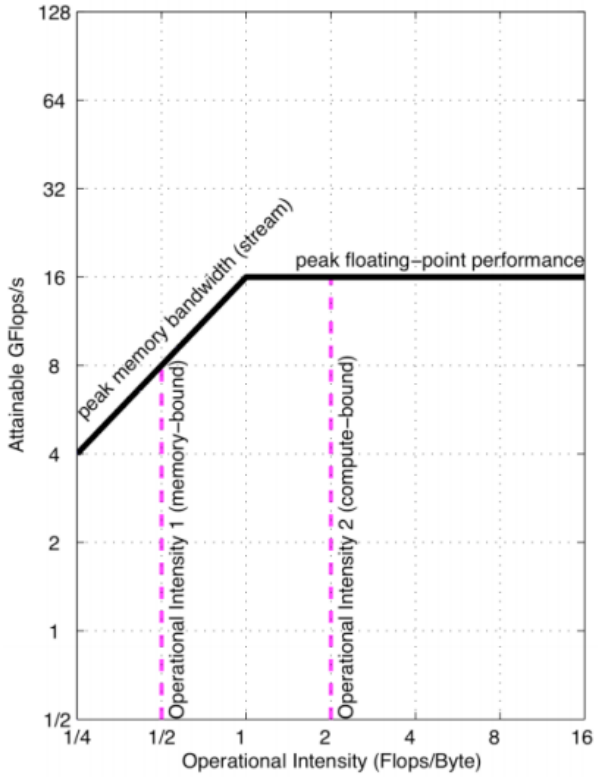


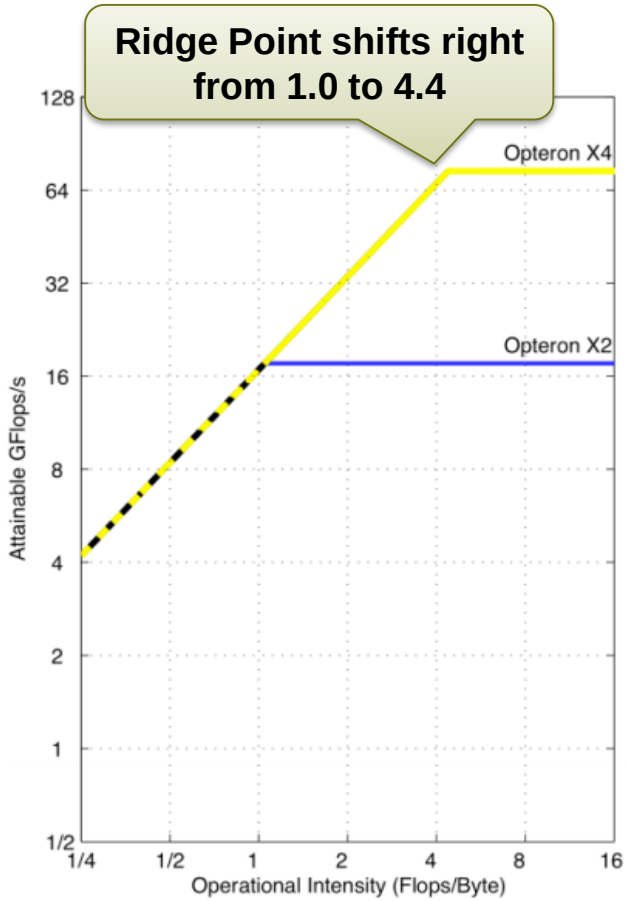
Measures the traffic between the caches and DRAM. But why?

Roofline Model

Attainable GFlops/sec = Min(Peak Floating Point Performance,
Peak Memory Bandwidth x Operational Intensity)

- **A kernel with a given OI lies somewhere in the vertical line with x=OI**
- **Ridge point:** intersection of the diagonal and horizontal roof
  - *Its x-coordinate is the minimum operational intensity required to achieve maximum performance*
  - *It suggests the level of difficulty for programmers and compiler writers to achieve peak performance*



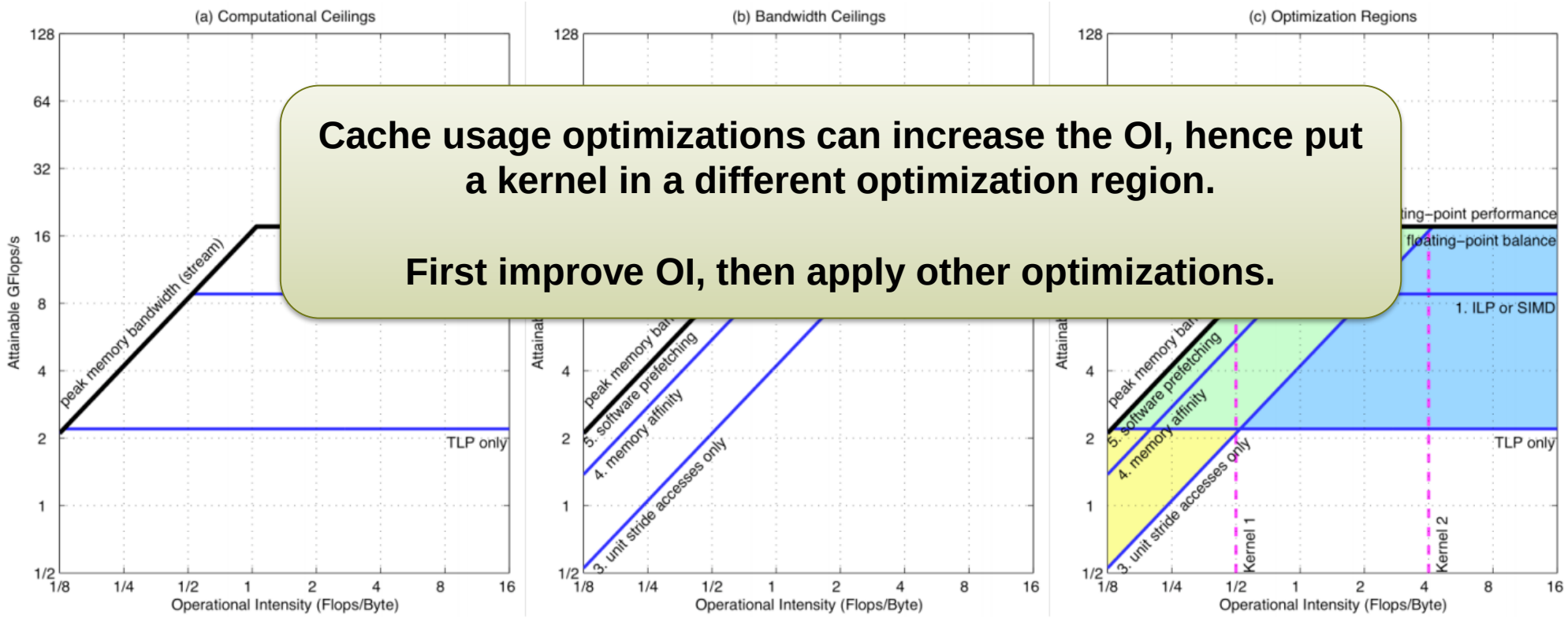


Opteron X4:

- Can issue 2 FP SEE2 instructions per cycle
- Slightly faster clock rate
- >4x gain in peak performance w.r.t. X2

Adding Ceilings

- **What if your program is far from the roofline?**
  - Ceilings can help us: you cannot break through a ceiling without performing the associated optimization.

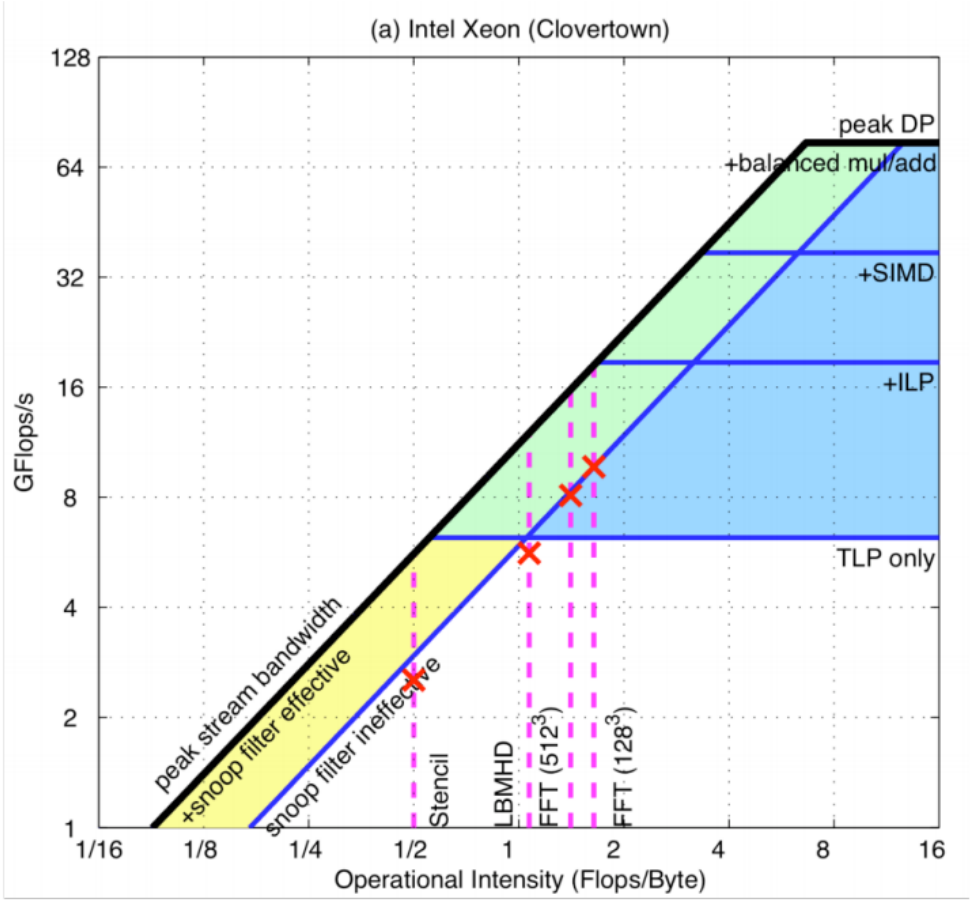


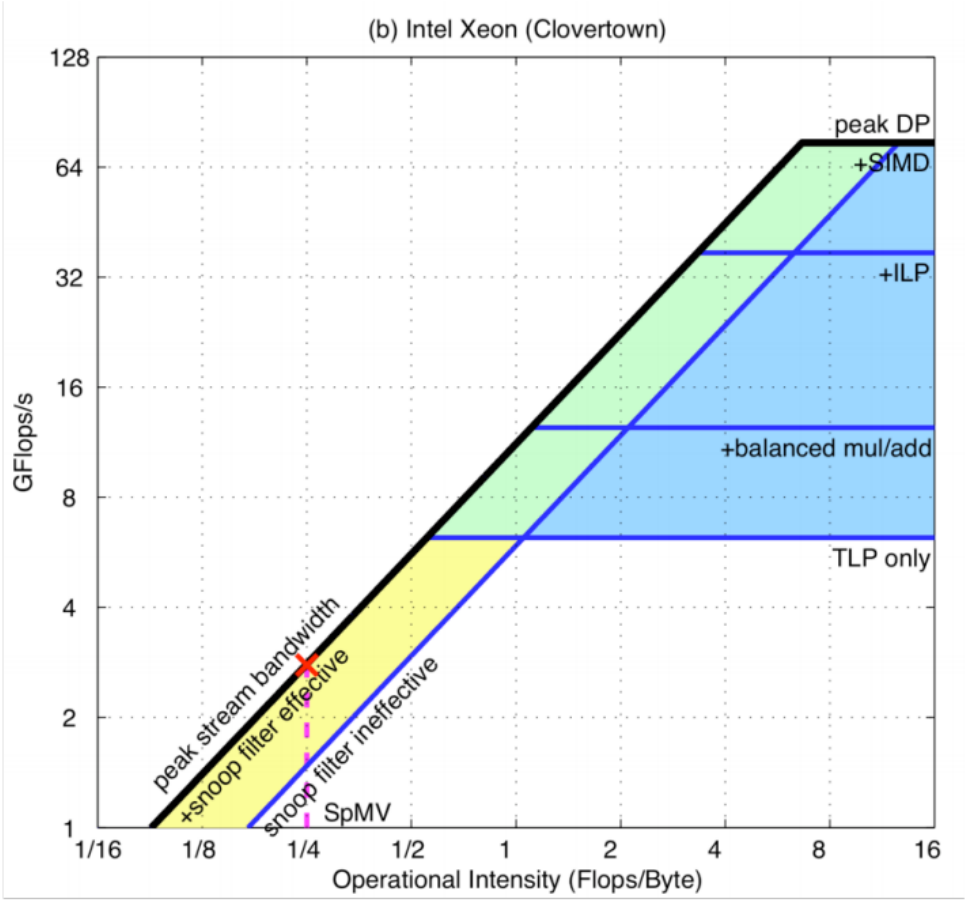
Cache usage optimizations can increase the OI, hence put a kernel in a different optimization region.

First improve OI, then apply other optimizations.

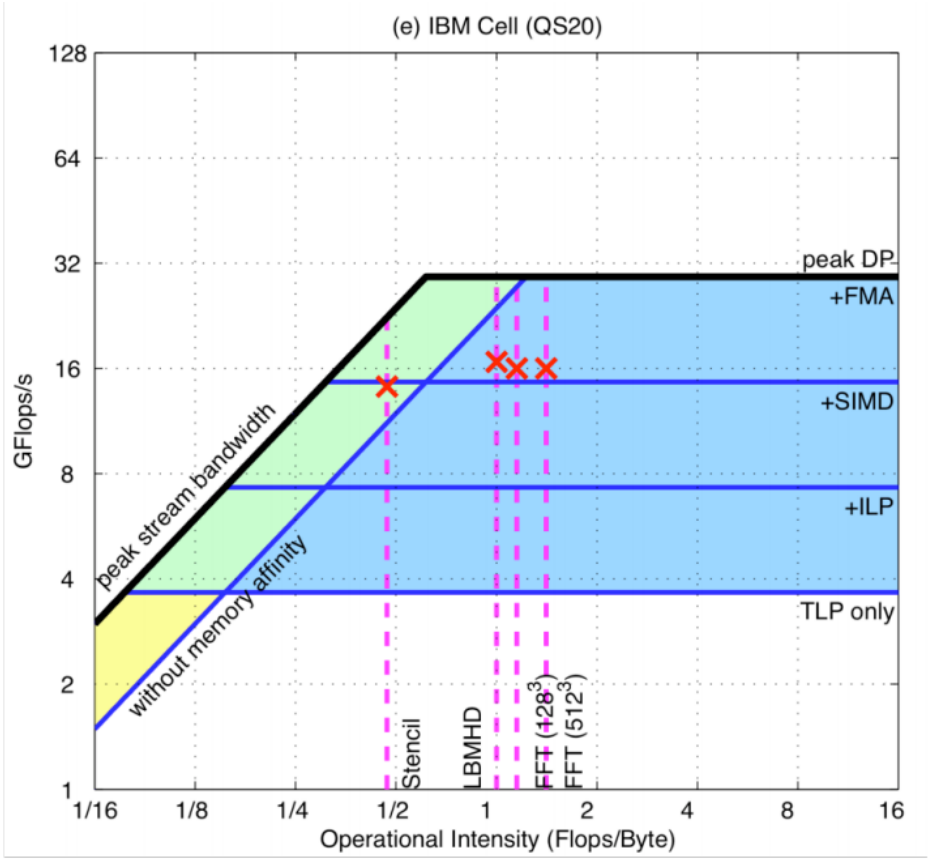
  - The height of the gap between a ceiling and the next higher one is the potential reward for trying that optimization
  - Their order suggests the optimization order. Lower ceilings: easy to implement by the programmer or likely realized by the compiler.

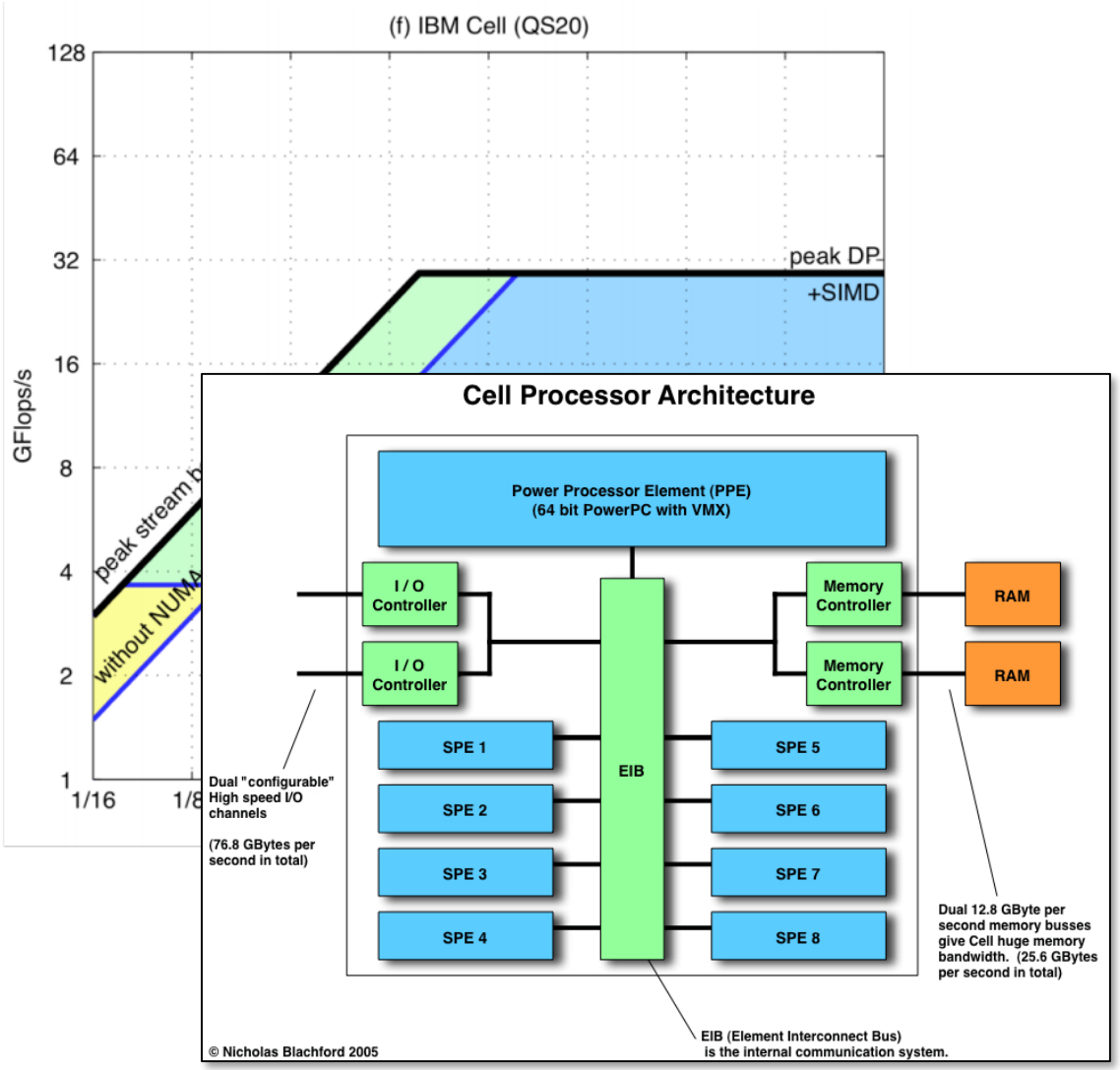
Models & Results



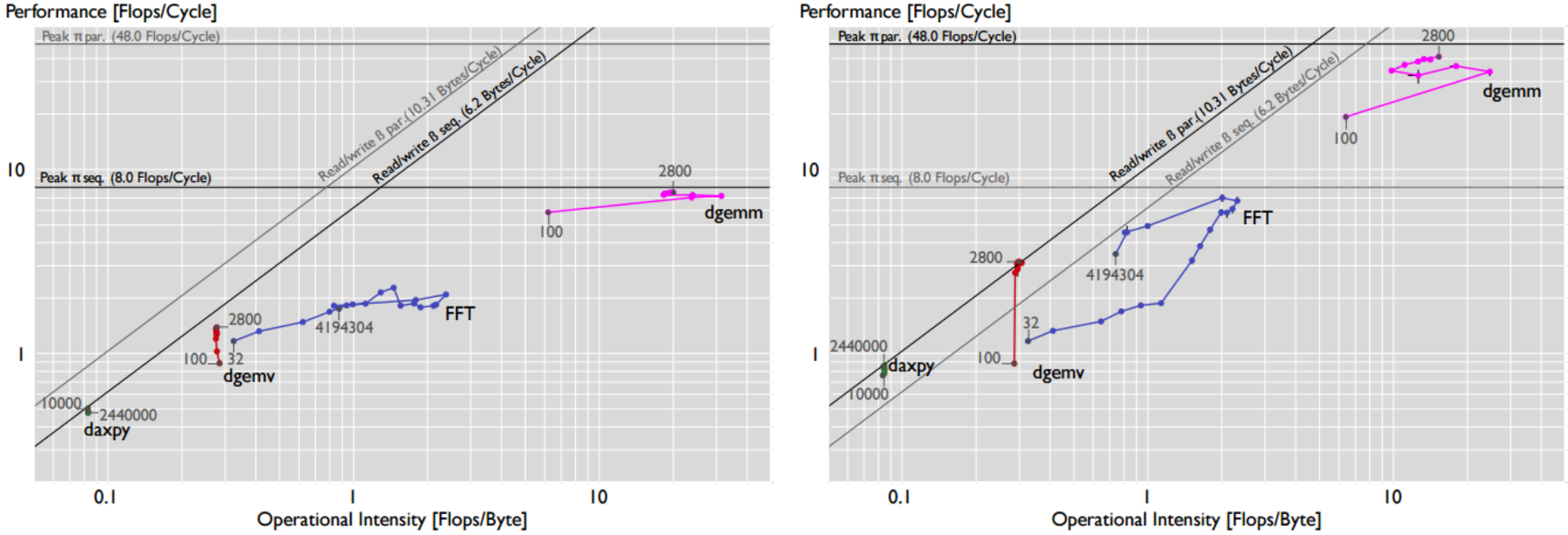


Models & Results





Multithreading



- **The ridge point shifts from 1.3 to 4.6**
- **Increasing the input makes parallelization gain efficiency**
  - Until when the working set gets too big to stay in cache

Ofenbeck, Georg, et al. "Applying the roofline model." Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on. IEEE, 2014.

Applying the Roofline Model

- **For each kernel, we need to measure:**
  - **The work W**
    Counters for floating point operations

$$W = \text{Scalar\_double} + \text{SSE\_double} \times 2 + \text{AVX\_double} \times 4$$

E.g., W on a Sandy Bridge platform

  - **The runtime T**
    Read Time Stamp Counter (RDTSC) is still a right choice
  - **The memory traffic Q**
    LLC misses can be an underestimation
    Measure raw traffic on the memory controller if possible (i.e.,Intel PCM)

- **For each architecture, we need to measure:**
  - The peak performance π: microbenchmarks or manual
  - The memory bandwidth β: microbenchmarks, most challenging

LibLSB: https://spcl.inf.ethz.ch/Research/Performance/LibLSB/

Ofenbeck, Georg, et al. "Applying the roofline model." Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on. IEEE, 2014.