Performance Analysis

Simon McIntosh-Smith

simonm@cs.bris.ac.uk

University of BRISTOL

Introduction

In HPC we care about performance.

It's easy to make a code go faster, but how do you know when it's really performing **well**?

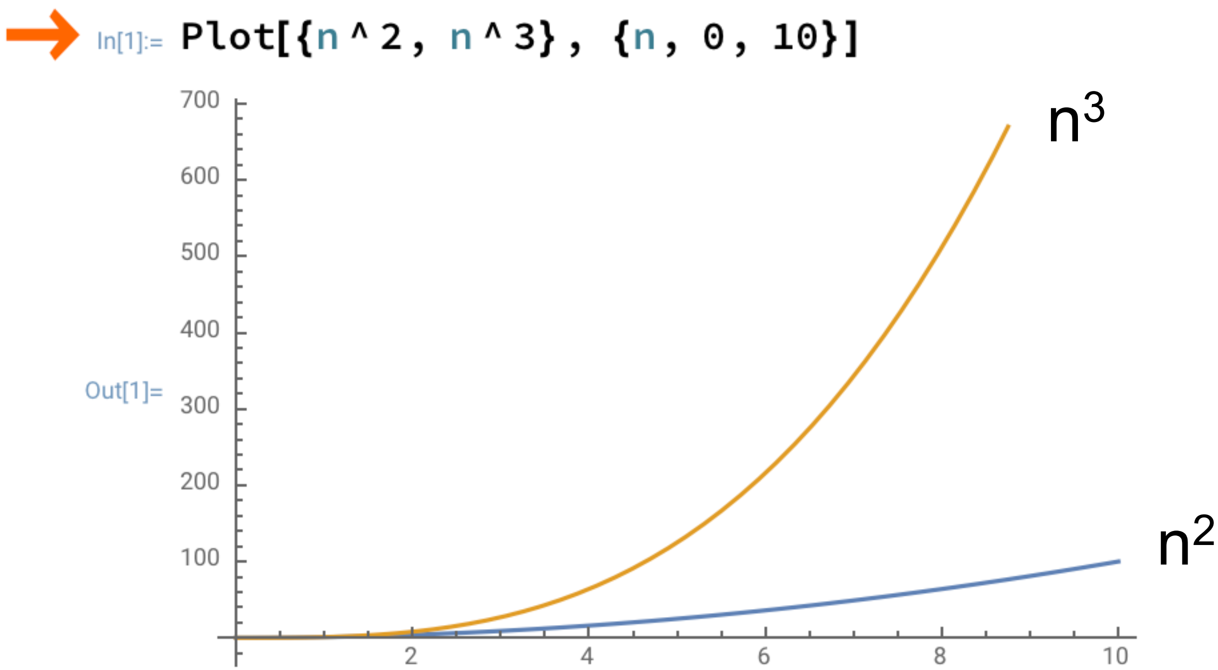
We need ***performance analysis***

Performance analysis

- Understand the characteristics of the algorithm
  - What is the overall algorithmic complexity?
    - For compute?
    - For data movement?

- E.g. vector-vector and vector-matrix operations are O(n) for both compute and data movement, but matrix-matrix multiply is $O(n^3)$ compute to $O(n^2)$ data movement

$O(n^2)$ vs $O(n^3)$



- Implication: matrix-matrix multiply becomes compute-bound at big enough *n*, but all vector-vector or vector-matrix operations remain memory bandwidth bound

What is your rate limiting factor?

- Most HPC codes are
  memory bandwidth bound
- A few are **compute bound**
- Other possibilities:
  - Network bound (e.g. MPI communication)
  - I/O bound (e.g. writing to disk)
  - Memory latency bound
  - Memory capacity bound
  - …

Which one am I?

- Try to reason about this on paper first
- Remember: floating point and integer operations are generally very cheap
  - O(1) cycle for most int / float / double operations
  - Exceptions: divide, transcendentals (sin, cos), exp, log
  - Load/store 2-3 times slower in the best case (hitting L1 cache), can be much slower if missing the cache
- You could simply count bytes loaded and stored vs. floating point operations…

Example: Jacobi

```
for (row = 0; row < N; row++) {
  dot = 0.0;                                    O(N)
  for (col = 0; col < N; col++) {
    if (row != col)                             O(N2)
      dot += A[row + col*N] * x[col];
  }
  xtmp[row] =
    (b[row] - dot) / A[row + row*N];
}
```

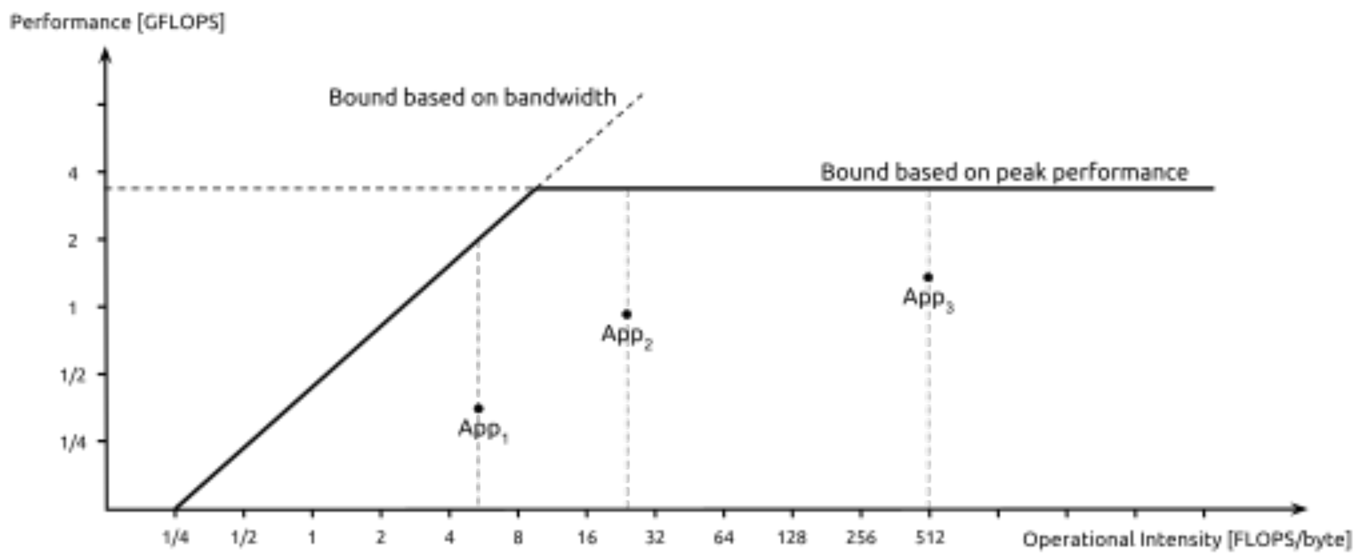

University of BRISTOL

Analysis Process

1. Think about the algorithm and select the best one you can
   – E.g. $O(n\log n)$ beats $O(n^2)$ etc.
2. Establish the rate limiting factor
   – E.g. memory bandwidth bound or compute bound
3. Optimise primarily for the rate limiting factor
4. Analyse the results to establish where to focus your efforts next
   – There are tools that can help you – profilers etc.

Roofline model

- A useful conceptual tool to establish whether compute bound or memory bandwidth bound



Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (April 2009), 65-76. DOI: https://doi.org/10.1145/1498765.1498785

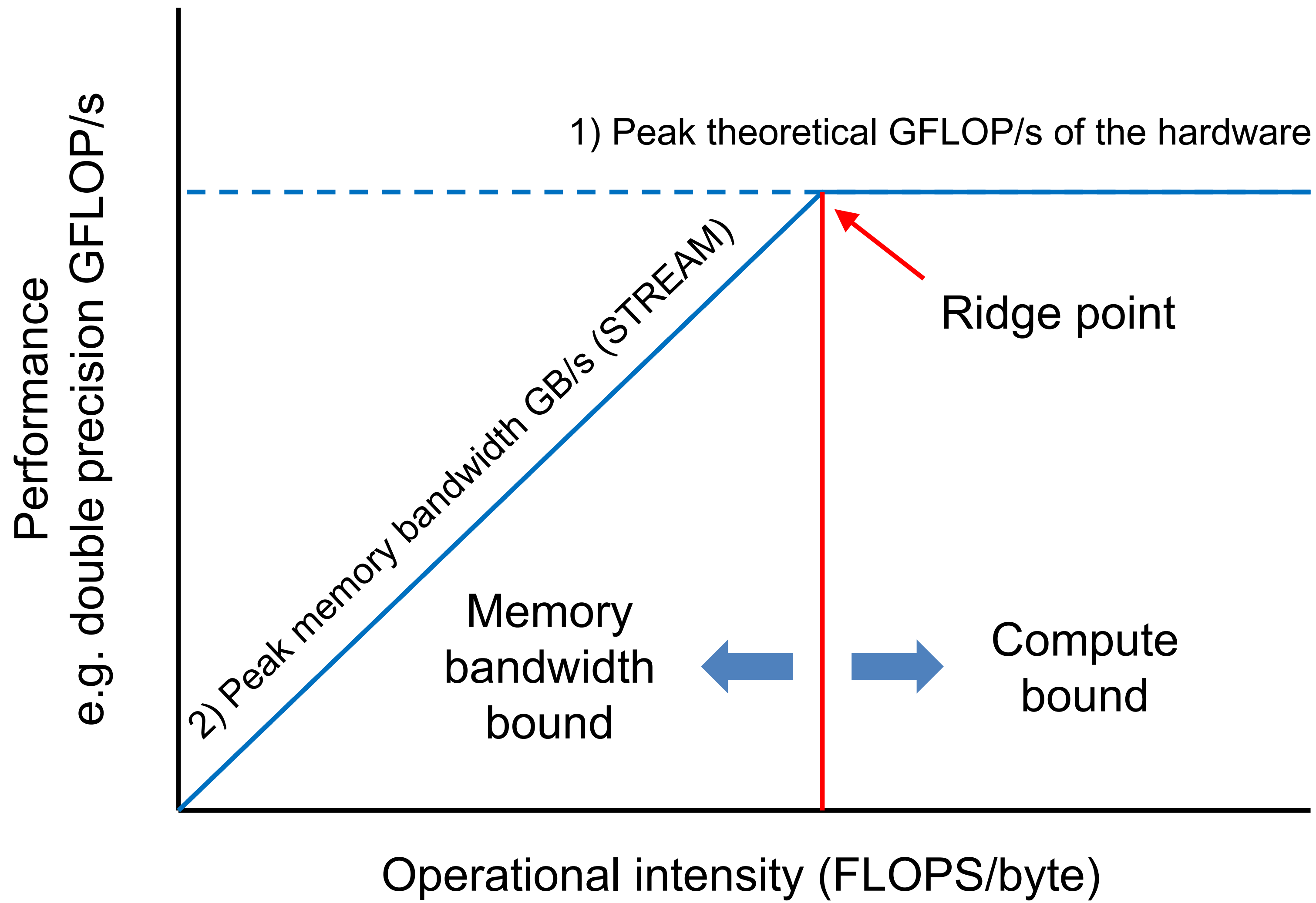
Some definitions

- **Operational Intensity (OI)**
  - Operations per byte of memory traffic
  - An operation could be floating point, integer …
  - Traffic is measured at main memory (DRAM)
    - i.e. the number of bytes transferred between the last level cache and memory, rather than between the processor and the caches
  - Also known as “arithmetic” or “computational” intensity

- Example:
  - **double a[], b[], c[];   a[i] += b[i] * c[i];**
  - 24 bytes loaded, 8 bytes stored, 2 operations (+, *)
  - → Operational intensity of (ops)/(bytes) = 2/32 = 1/16

University of BRISTOL

A roofline graph



University of BRISTOL

Roofline definition

$$\text{Attainable GFLOP/s} = \mathit{min} \begin{cases} \text{Peak floating-point performance} \\ \text{Peak memory bandwidth} \times \text{Operational intensity} \end{cases}$$

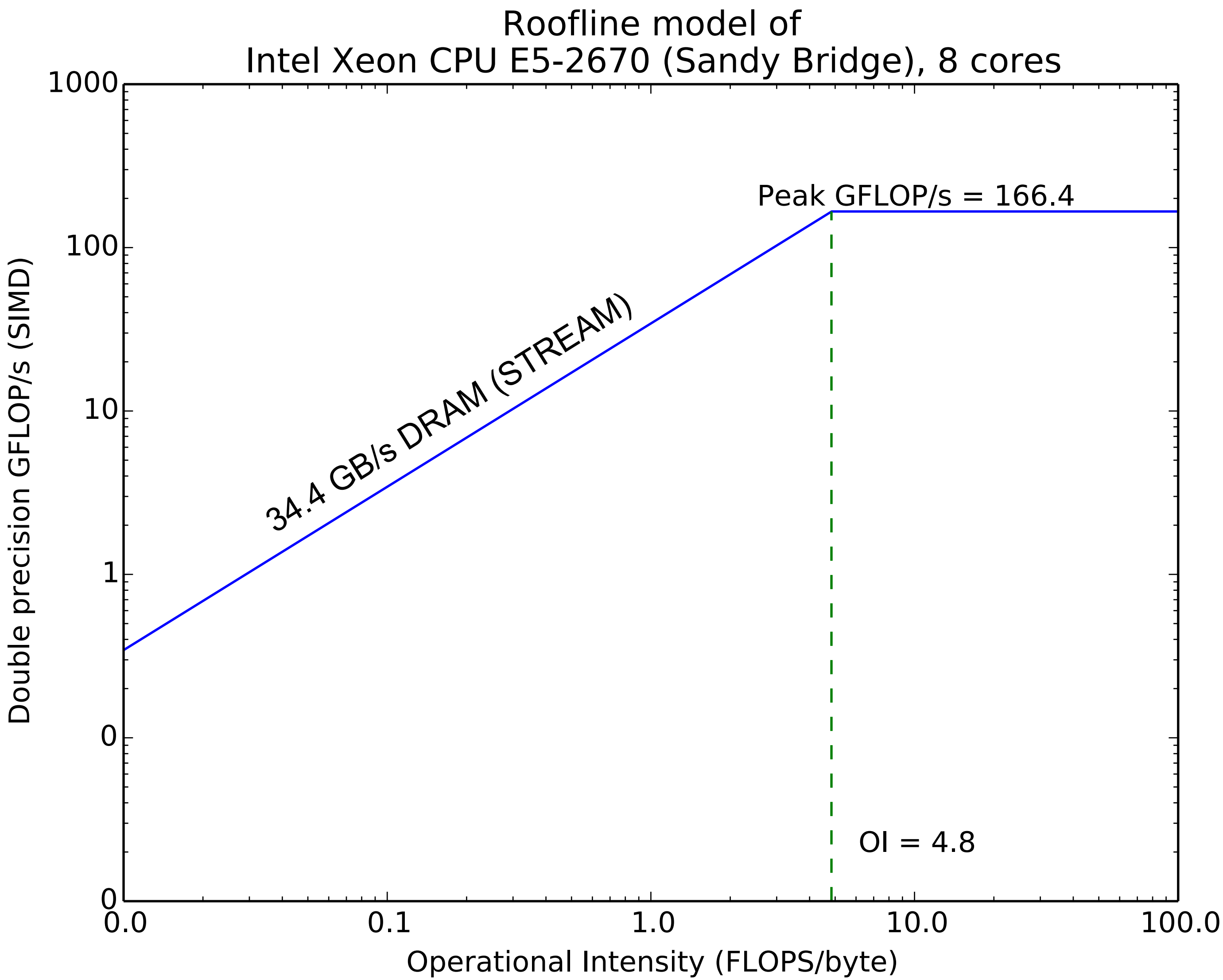
University of BRISTOL

The original definition is slightly dated

- It didn't model in-cache behaviour
- Vague about SIMD and multi-core

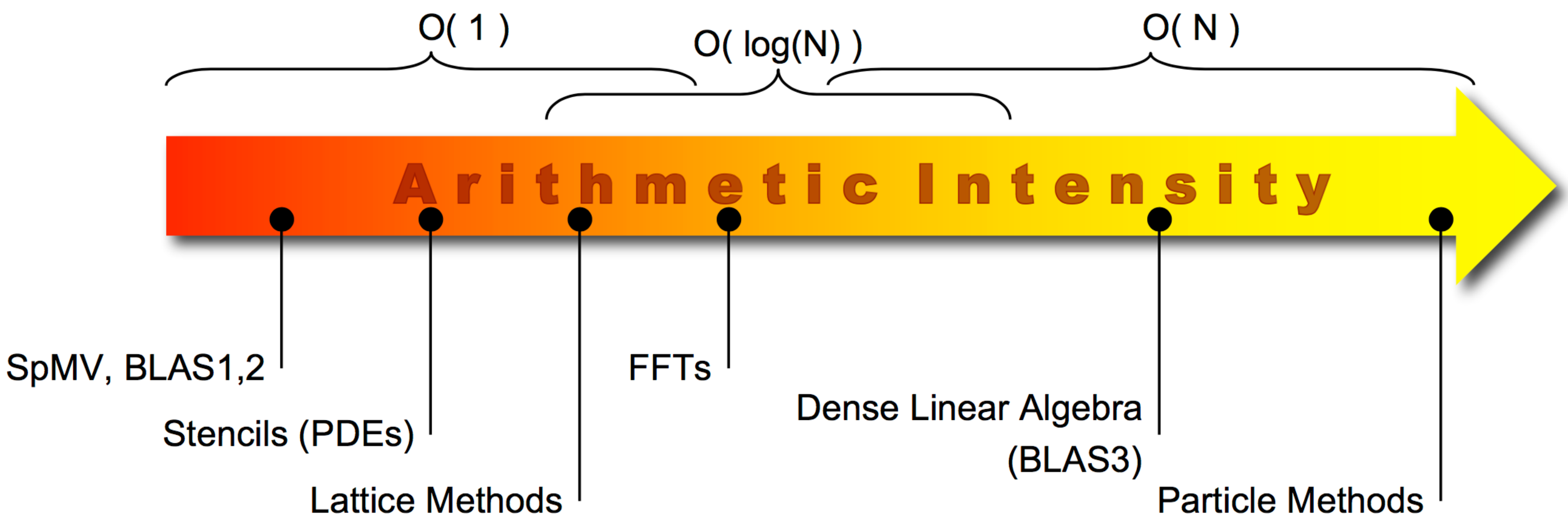
- "The *x*-coordinate of the ridge point is the minimum operational intensity required to achieve maximum performance."
  - But this implies that performance is all about FLOP/s, which it isn't
  - E.g. if you have a memory bandwidth bound code, performance is all about moving bytes. FLOP/s might be much less important

Roofline for BCp3



University of BRISTOL

Operational (Arithmetic) Intensity



- **True Operational Intensity (OI) ~ Total FLOPS / Total DRAM Bytes**
  – Constant with respect to problem size for many problems of interest
  – Ultimately limited by unavoidable traffic to DRAM
  – Reduced by conflict or capacity cache misses

Jacobi Operational Intensity

```
for (row = 0; row < N; row++) {
  dot = 0.0;                                    O(N)
  for (col = 0; col < N; col++) {
    if (row != col)                             O(N2)
      dot += A[row + col*N] * x[col];
  }
  xtmp[row] =
    (b[row] - dot) / A[row + row*N];
}
```

$OI_{fp64} = 2/16 = 0.125$
$OI_{fp32} = 2/8 \quad = 0.25$

When are you doing well?

- Be near your limit, whatever that is

- **For memory bandwidth bound codes:**
  - Aim to achieve a good fraction of measured STREAM bandwidth
    - STREAM is a simple memory bandwidth benchmark. It represents the maximum bandwidth you could ever achieve. Typically get to within 85% of the theoretical peak memory bandwidth of the hardware

When are you doing well?

- Be near your limit, whatever that is

- **For compute bound codes:**
  - Aim to achieve a good fraction of the theoretical peak floating point performance of the hardware
    - Is that for scalar or SIMD?
    - Single or double precision? (or even “half” now)

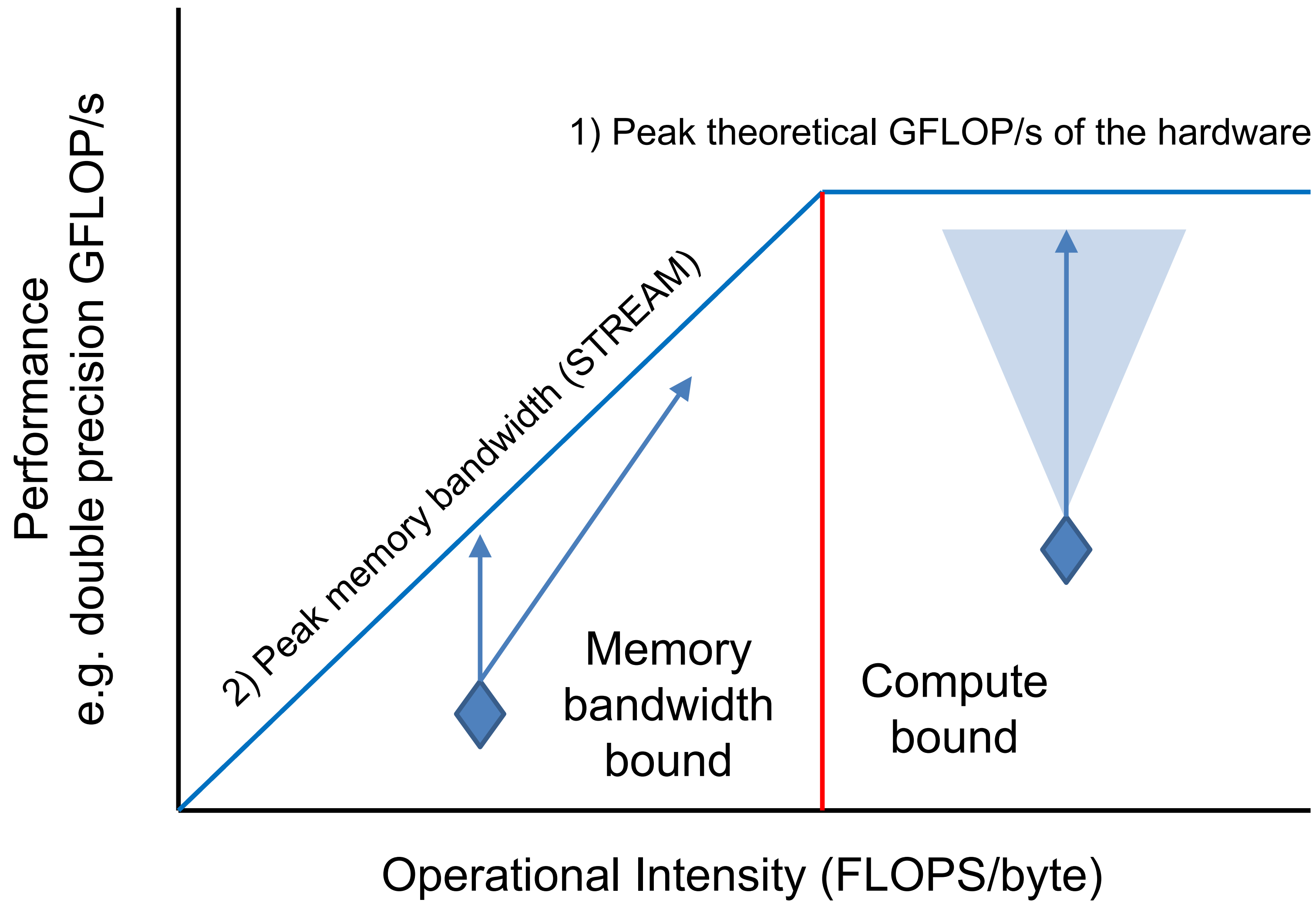
Use the right roofline

The following have big effects:

- Single core vs multi-core
- Scalar vs SIMD
- Single precision vs double precision

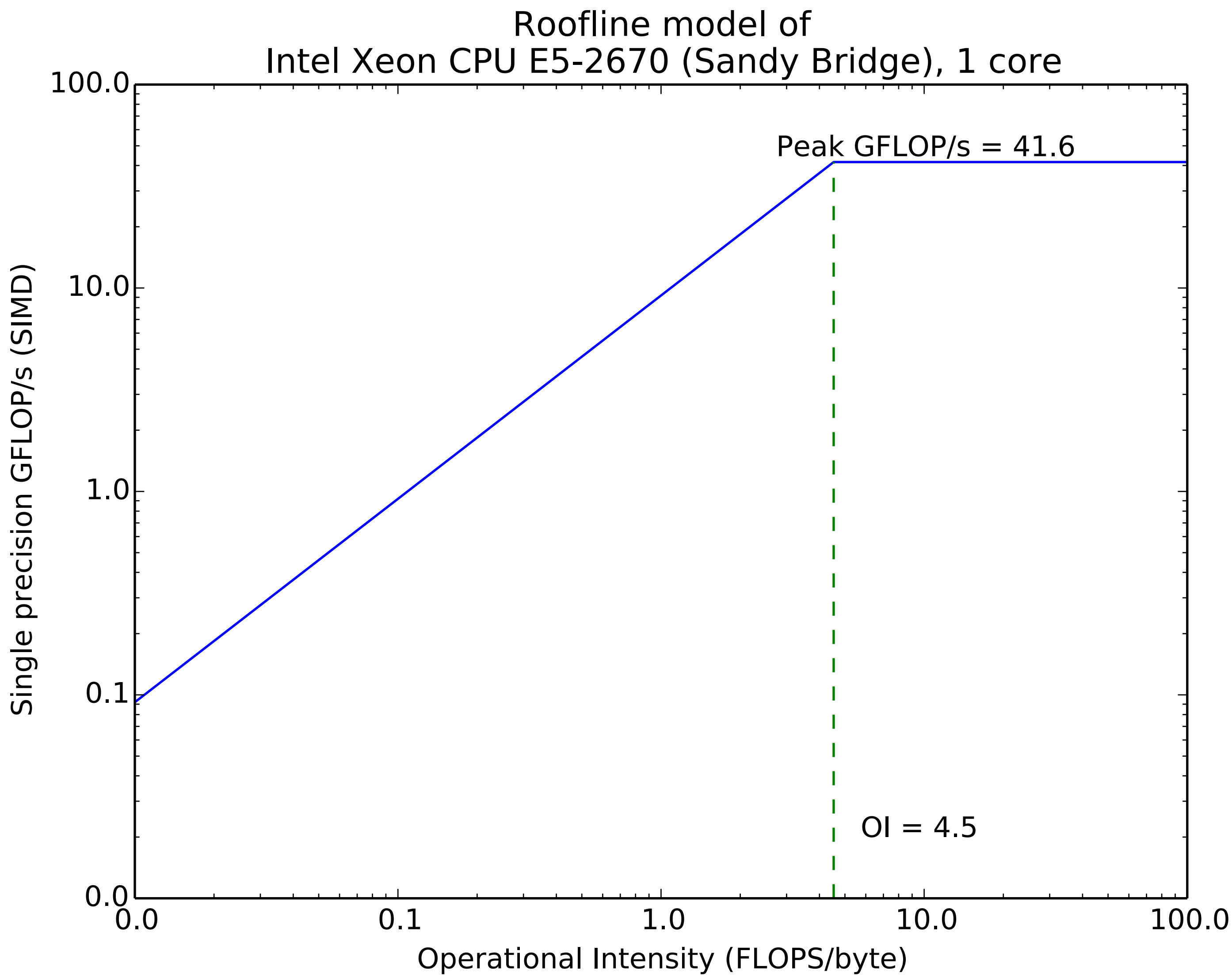
Use the right version of the graph for you

Improving performance as seen through the Roofline model



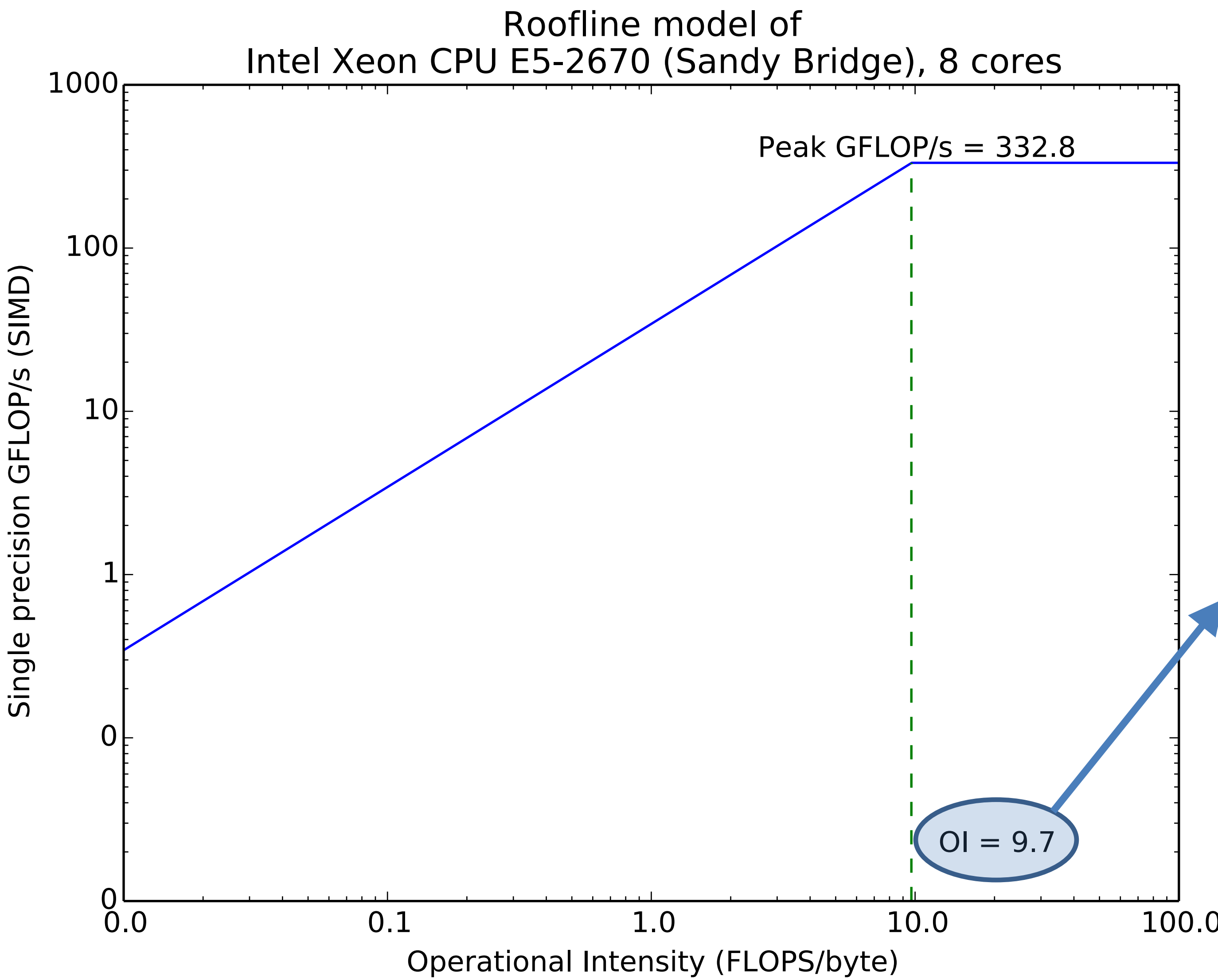
University of BRISTOL

For the scalar optimisation exercise



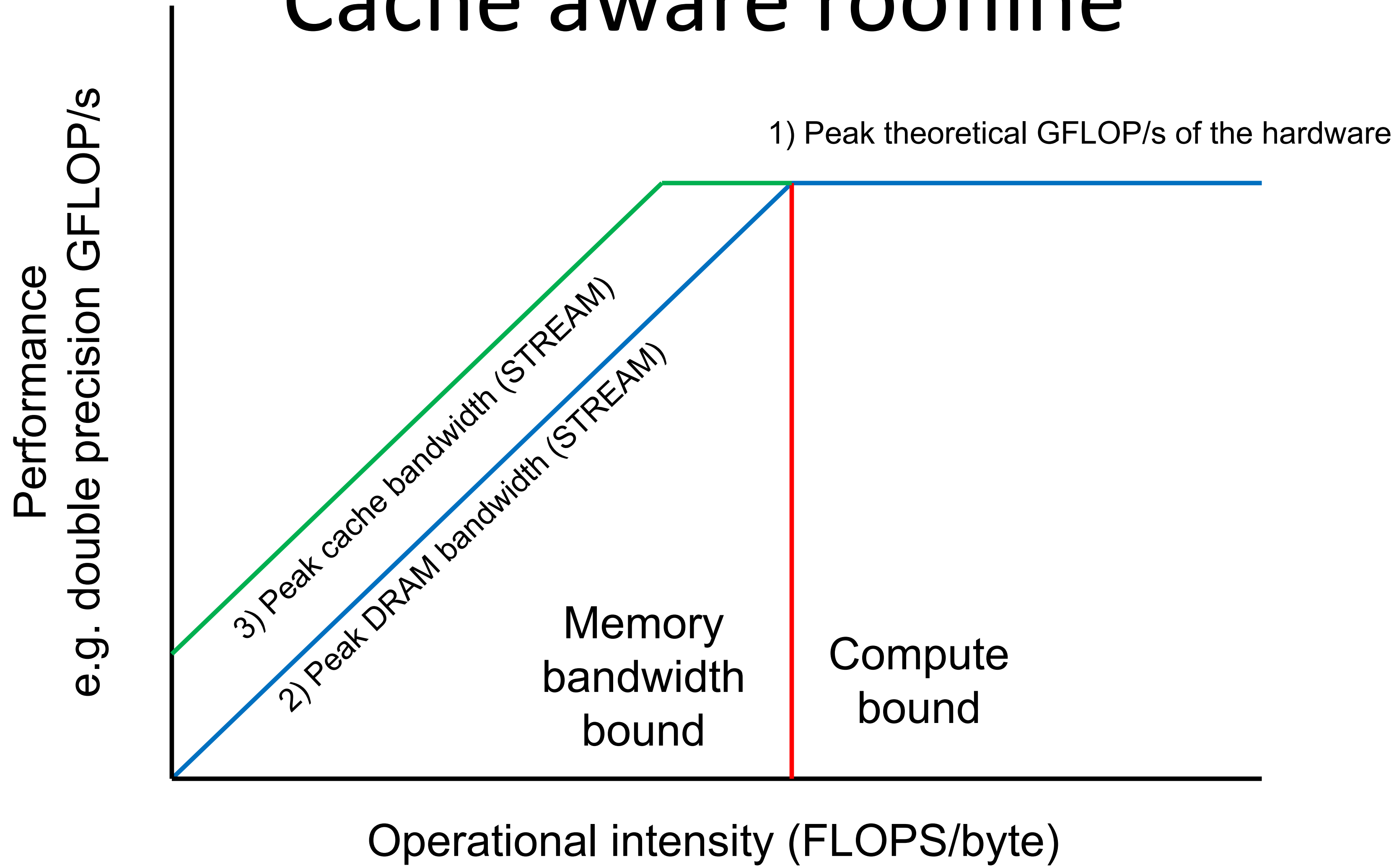
University of BRISTOL

Multicore raises the height of the roof



Single core OI was 4.5. Now 8 times as many core sharing the bandwidth. So why isn't this 8*4.5 = 36?

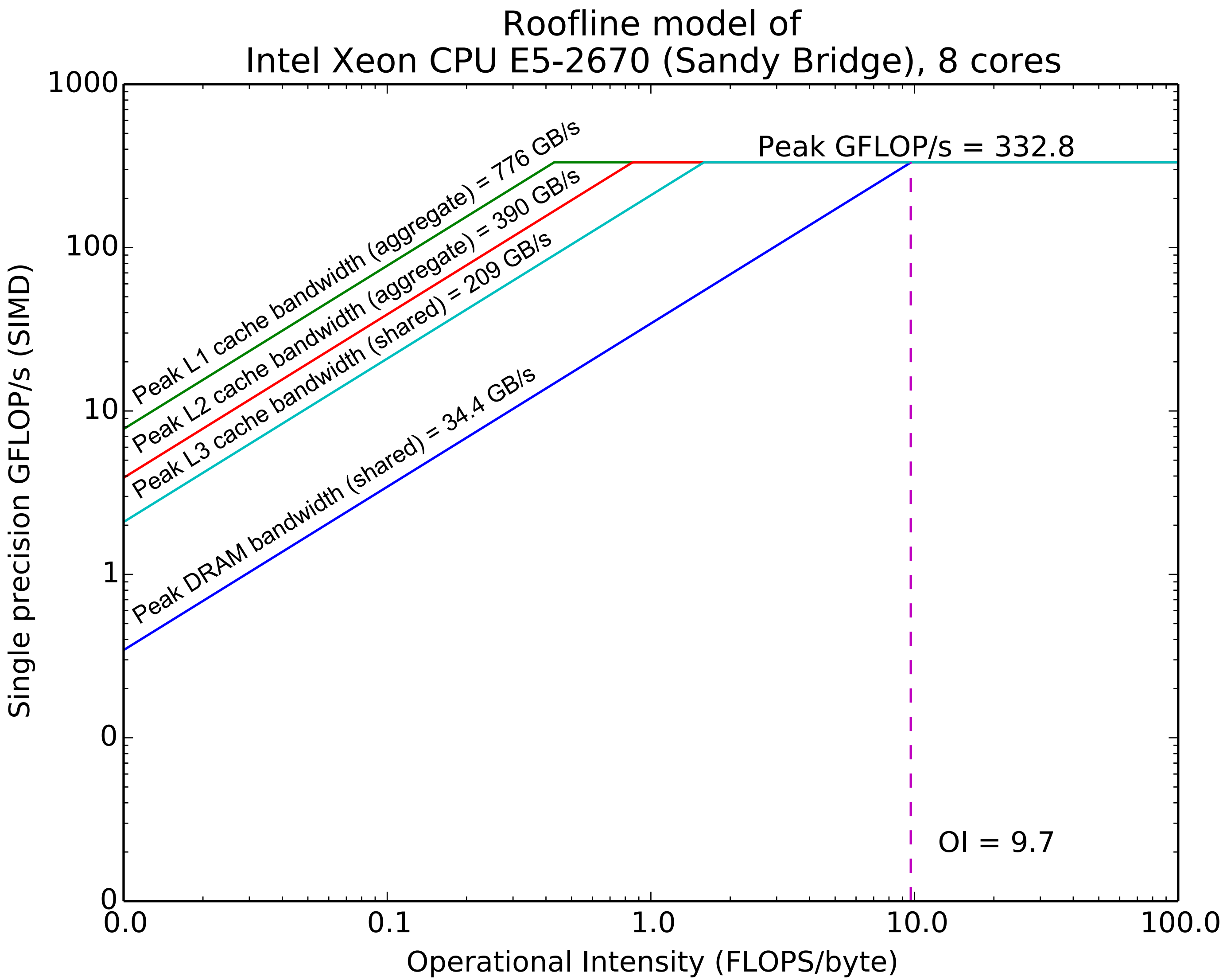
Cache aware roofline



Ilic, Aleksandar, Frederico Pratas, and Leonel Sousa. "Cache-aware roofline model: Upgrading the loft." *IEEE Computer Architecture Letters* 13.1 (2014): 21-24.

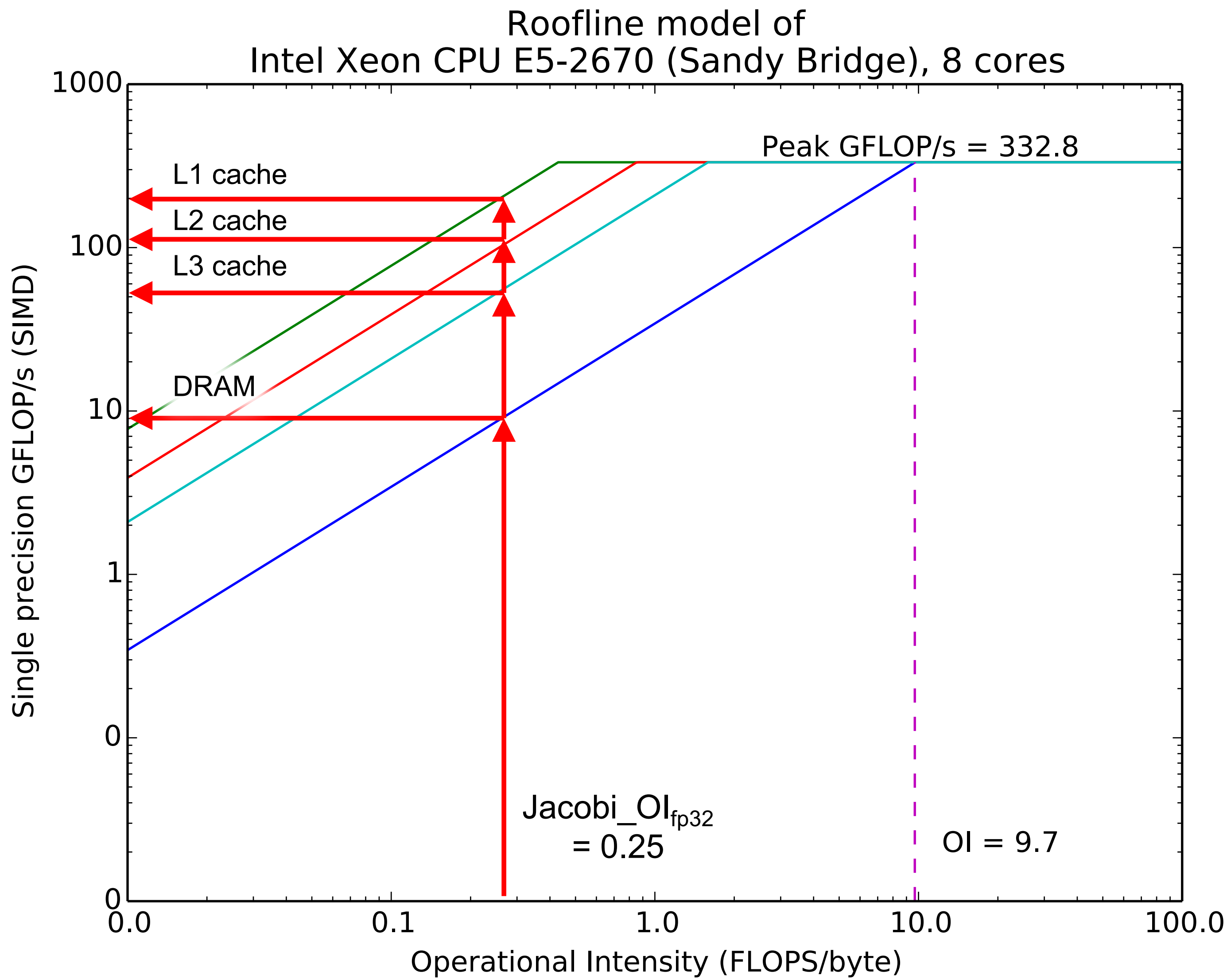
University of BRISTOL

Cache aware roofline of BCp3



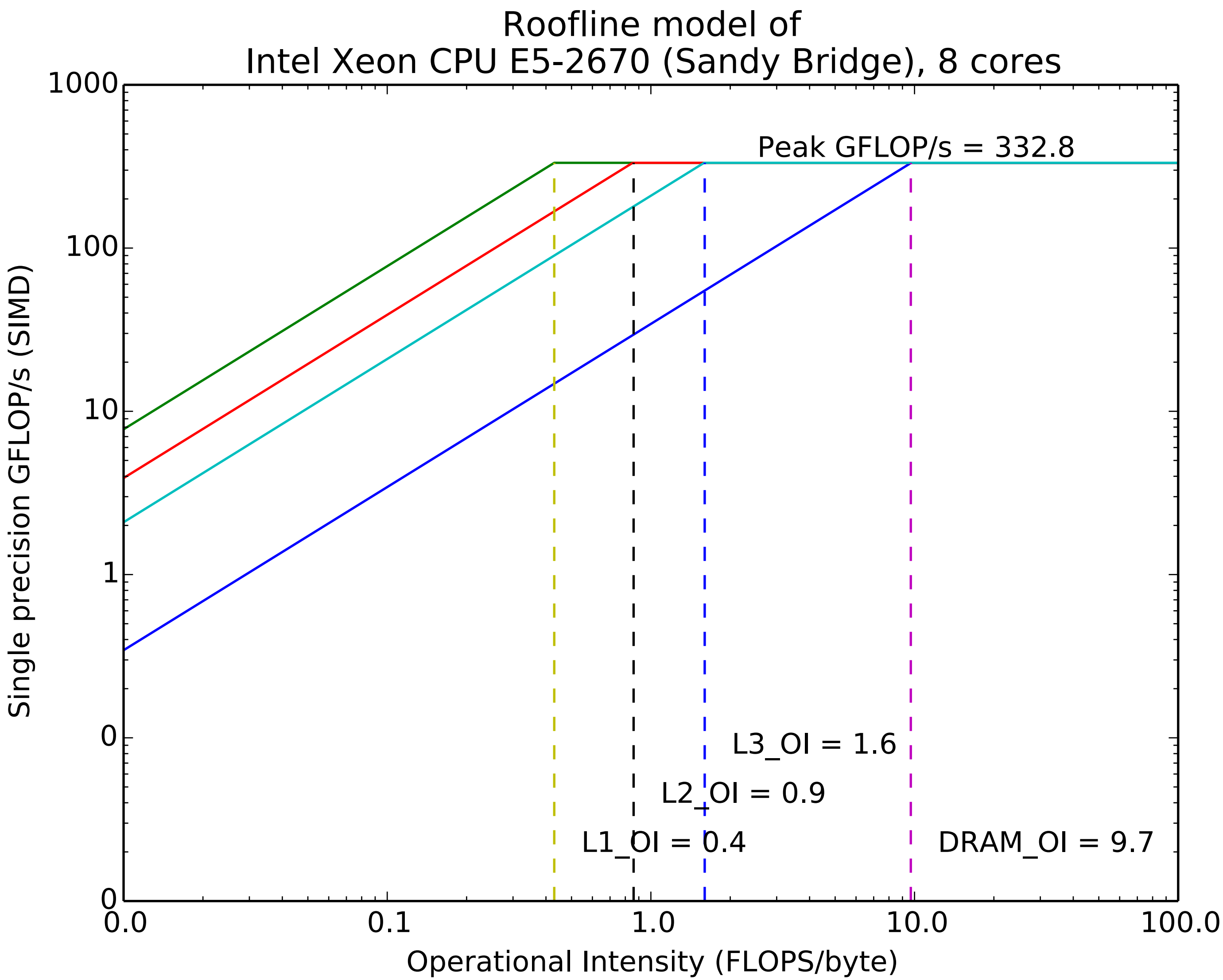
University of BRISTOL

Cache aware roofline of BCp3

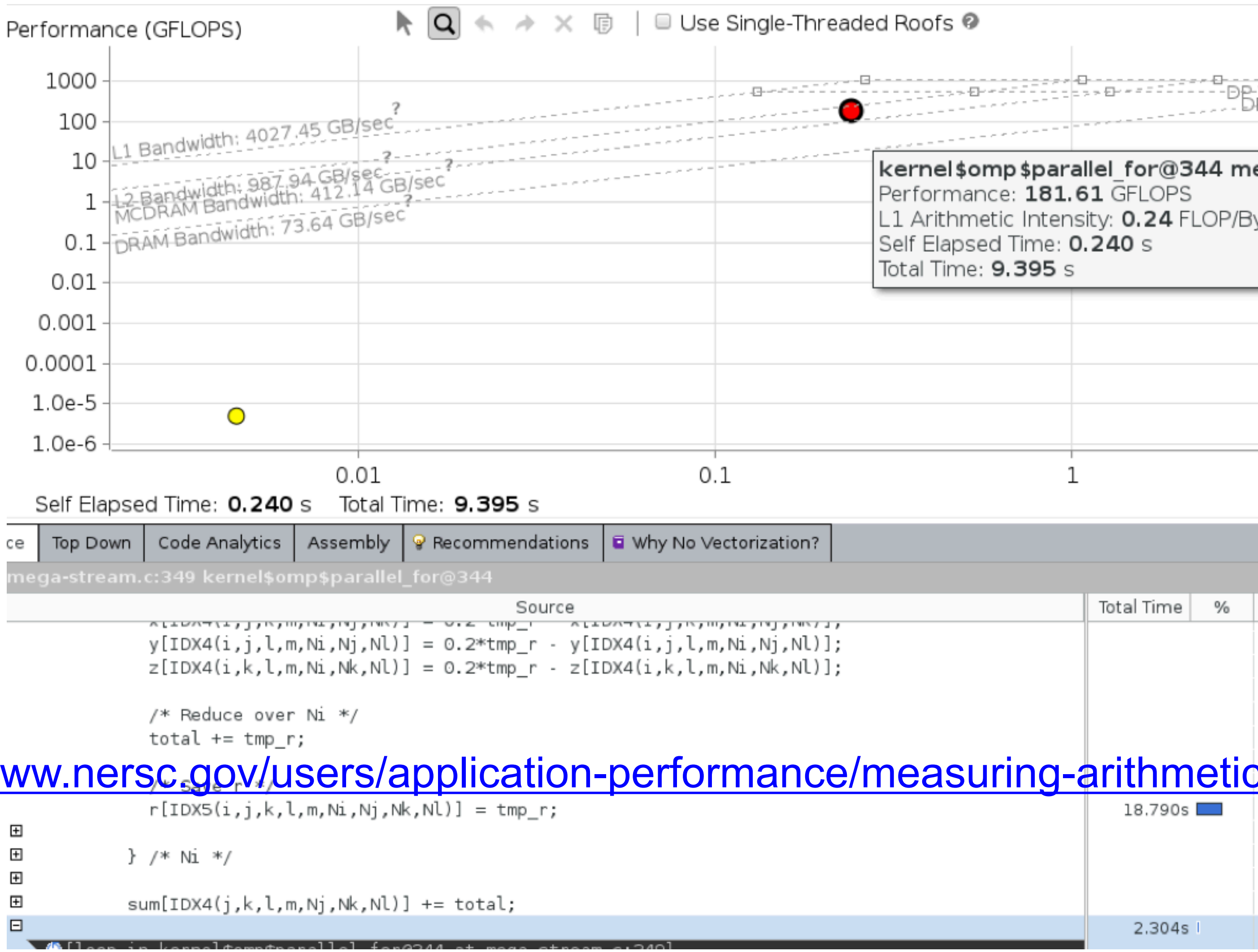


University of BRISTOL

OIs for each level of the memory



Example Intel Advisor roofline output



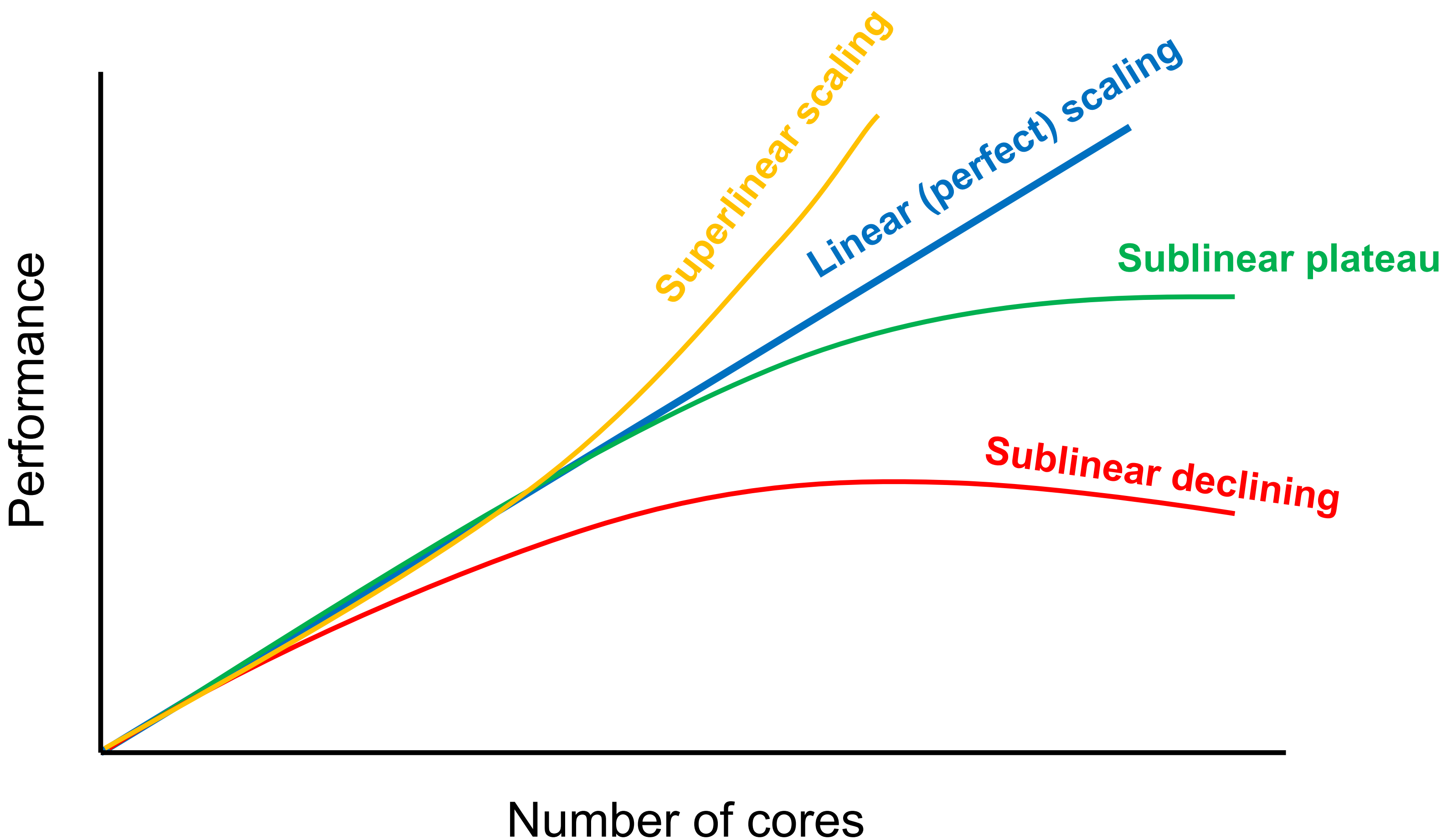
See:

http://www.nersc.gov/users/application-performance/measuring-arithmetic-intensity/

University of BRISTOL

What else might we be interested in?

For parallel programs, such as OpenMP: scaling



Summary

- It's not enough to just make a program faster than it was
- You need to quantify how fast your program is, relative to what is possible
- The roofline model is one useful way to measure and assess what fraction of peak you are achieving
- Consider how close you are to linear scaling when going parallel

Further reading

- The original roofline paper:
  Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (April 2009), 65-76. DOI: https://doi.org/10.1145/1498765.1498785
- The cache-aware extension to roofline:
  Ilic, Aleksandar, Frederico Pratas, and Leonel Sousa. "Cache-aware roofline model: Upgrading the loft." *IEEE Computer Architecture Letters* 13.1 (2014): 21-24.