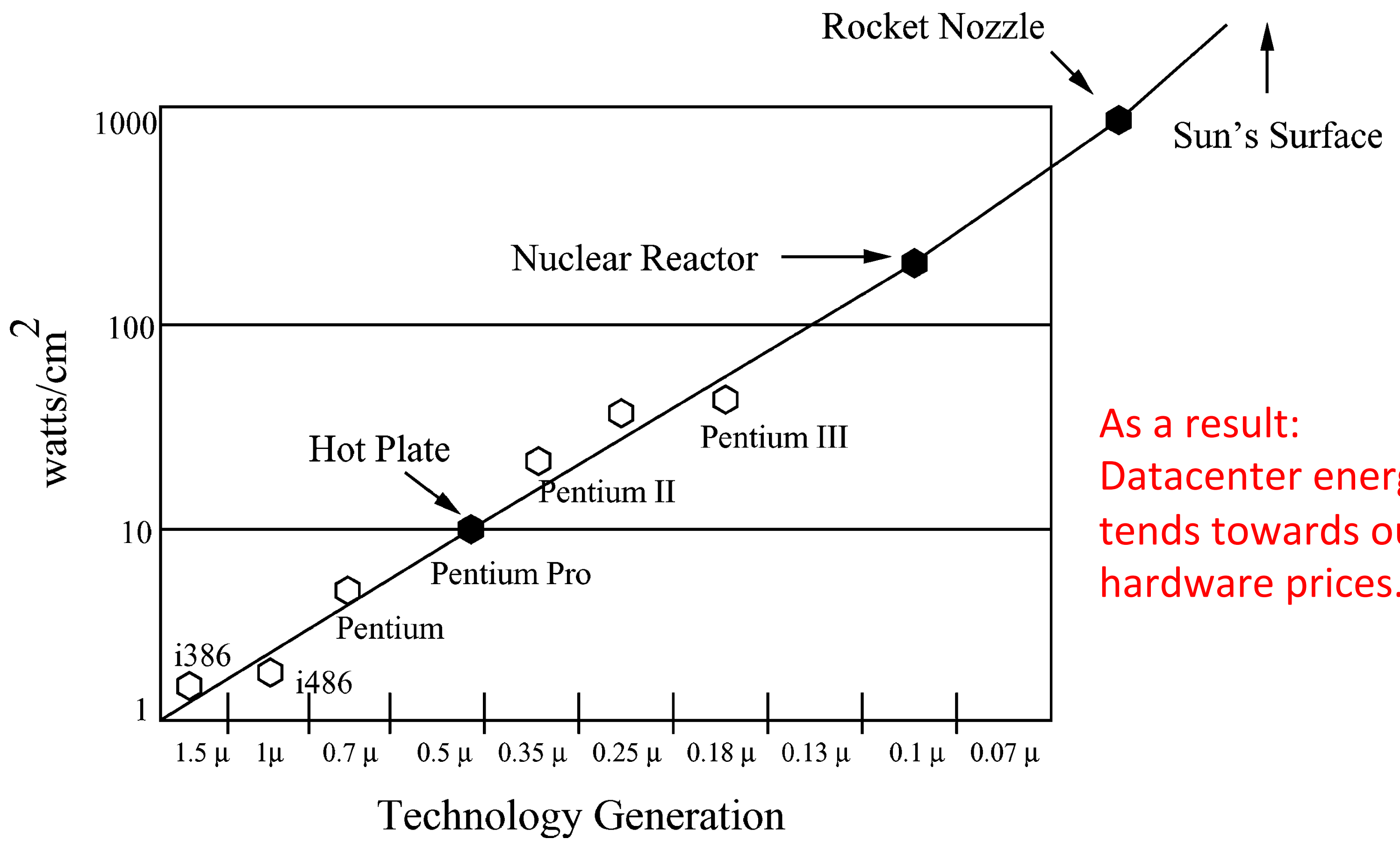
Power Consumption of Microprocessors and Roofline Model

Sarala Arunagiri

1 October 2013

Power consumption is a major factor that limits the performance of computer systems



As a result:
Datacenter energy costs tends towards outpacing hardware prices.

Increased power consumption => more electricity bill and also cost of cooling
Focus in this class is on

power consumption in the Microprocessor (CPU) and ways to reduce it.

Important Points to Note

The focus of this study is the power consumed by a microprocessor (chip level). This power consumption is of two types – dynamic and static. It is important to understand the factors affecting these two types of power consumption and how we could reduce them.

Power Reduction Techniques For Microprocessor Systems

- Reference: Vasanth Venkatachalam and Michael Franz. 2005. Power reduction techniques for microprocessor systems. ACM Computing Surveys 37, 3 (September 2005), 195-237. DOI=10.1145/1108956.1108957 http://doi.acm.org/10.1145/1108956.1108957

- For the class we use slides (1 to 16) of the presentation by Milo Martin and Amir Roth on Power and Energy as part of the course on Computer Organization and Design at University of Pennsylvania.

  https://www.cis.upenn.edu/~milom/cis371/lectures/13_power.pdf

Roofline Model

Motivation

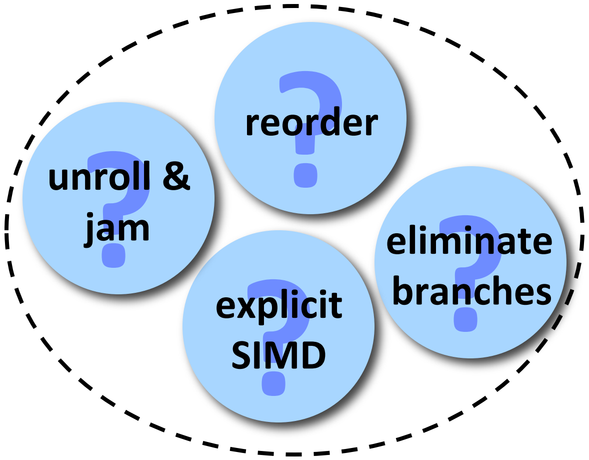
There are a variety of Multicore processors with diverse architectural features. This is due to the fact that there is no single architecture that is proven to perform well for all classes of kernels.

Problem: When one optimizes a kernel for a multicore architecture, how can we tell if it is good enough? How do you evaluate your level of optimization?

Optimization Categorization

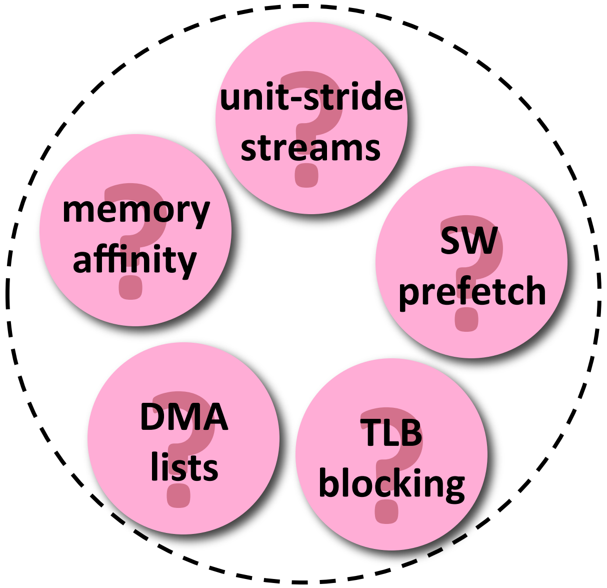
Maximizing In-core Performance

- **Exploit in-core parallelism (ILP, DLP, etc…)**
- **Good (enough) floating-point balance**



Maximizing Memory Bandwidth

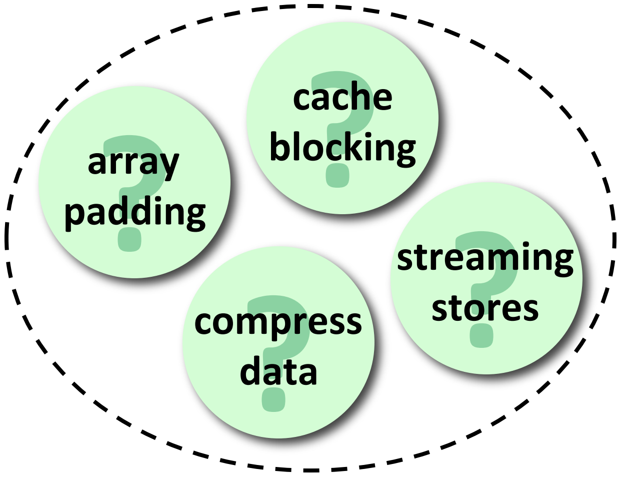
- **Exploit NUMA**
- **Hide memory latency**
- **Satisfy Little's Law**



Minimizing Memory Traffic

Eliminate:

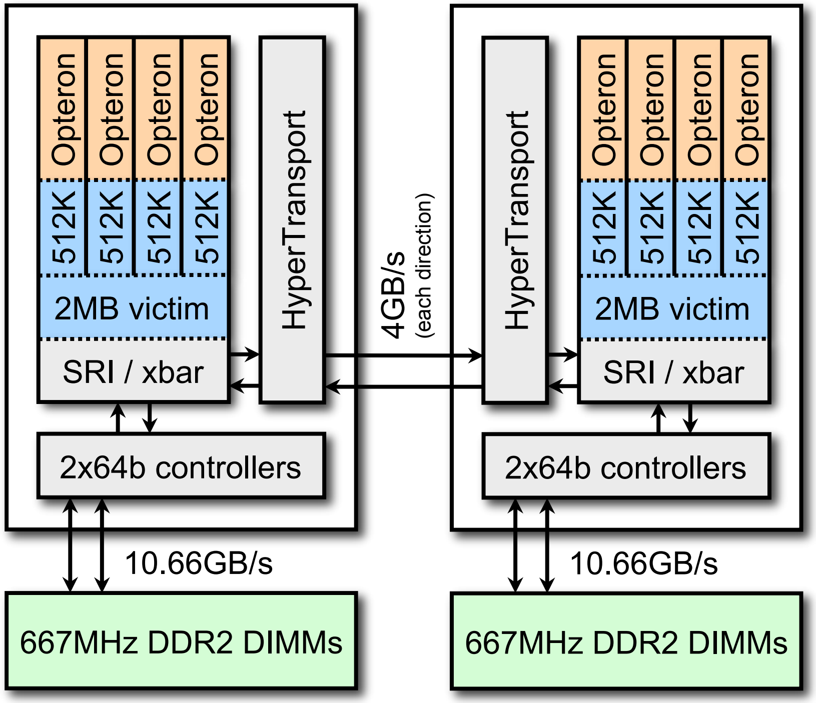
- **Capacity misses**
- **Conflict misses**
- **Compulsory misses**
- **Write allocate behavior**



Source: Samuel William's presentation on the roofline model.

Example Architecture - 1

- 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

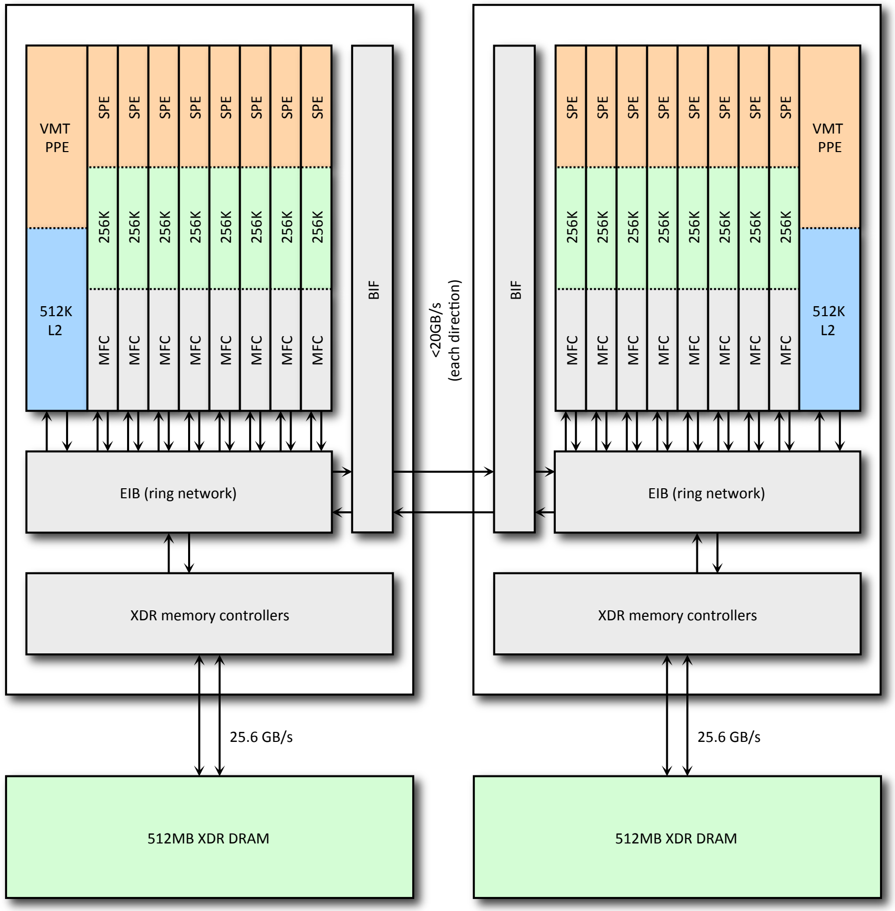


Source: graphics from Samuel William's presentation on the roofline model.

Example Architeture - 2

IBM QS20 Cell Blade

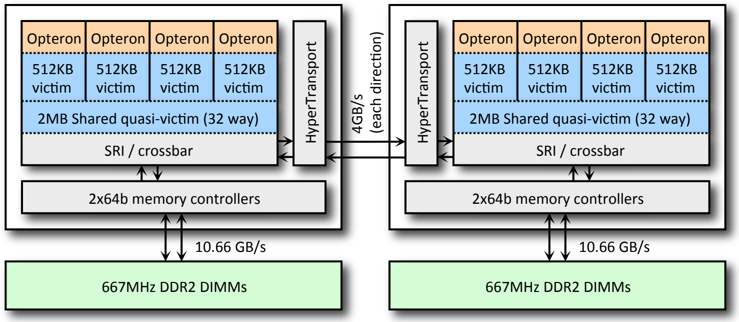
- Dual socket
- Two cell processors at 3.2 GHz
- 8 cores per cell processor (16 total)
- Two Gigabit Ethernet ports
- 1GBXDRAM memory (512 MB per processor)
- 410 Gflops peak performance



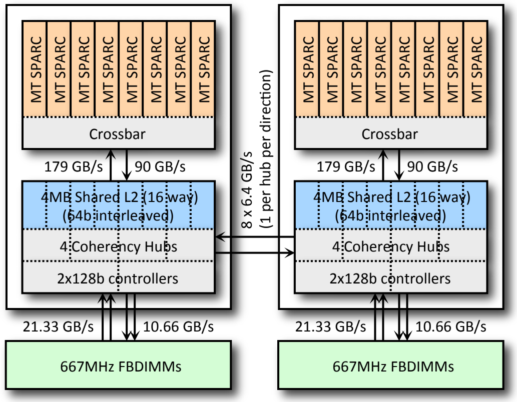
Source: graphics from Samuel William's presentation on the roofline model.

Many More Architectures

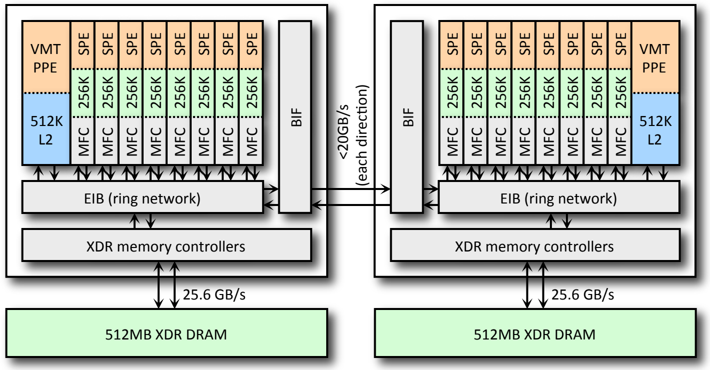
AMD Barcelona



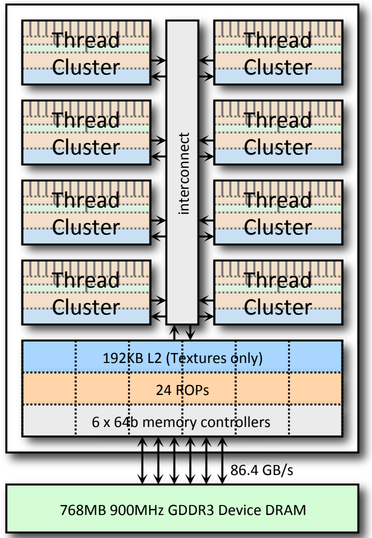
Sun Victoria Falls



IBM Cell Blade



NVIDIA G80



Source: graphics from Samuel William’s presentation on the roofline model.

Challenges / Goals

- We have extremely varied architectures.
- Moreover, the characteristics of numerical methods can vary dramatically.

- The result is that performance and the benefit of optimization can vary significantly from one (architecture x kernel) combination to the next.

- We wish to understand whether or not we've attained good performance (high fraction of a theoretical peak)
- We wish to identify performance bottlenecks and enumerate potential remediation strategies.

Source: Samuel William's presentation on the roofline model.

From the Horse’s Mouth

As part of the Google TechTalk “The Parallel Revolution Has Started: Are You Part of the Solution or Part of the Problem?” David Patterson presents an introduction to the Roofline Model. Please listen to that part of the talk from minute 40 to minute 53 of the video that can be accessed at -

http://www.youtube.com/watch?v=A2H_SrpAPZU

An Exercise Before Examining Illustrations of Usage of Roofline Model

Goal: Familiarize ourselves with the concept of operational intensity and learn to draw the roofline model.

Exercise

Exercise 1: What is the operational intensity (Ops/byte) of the following code segment, assuming each data is 4 bytes long? A, B, and C are vectors of complex numbers.

```
Code segment 1
For (i=0; i<300; i++){
    C_re[i] = A_re[i]*B_re[i] – A_im[i]*B_im[i];
    C_im[i] = A_re[i]*B_im[i] + A_im[i]*B_re[i];
}
```

Exercise

Exercise 1: What is the operational intensity (Ops/byte) of the following code segment, assuming each data is 4 bytes long?

```
Code segment 1
For (i=0; i<300; i++){
      C_re[i] = A_re[i]*B_re[i] – A_im[i]*B_im[i];
      C_im[i] = A_re[i]*B_im[i] + A_im[i]*B_re[i];
}
```

Answer:

Total number of operations in the segment =6*300

Number of reads = 4*300

Number of writes = 2*300

Number of operations per read/write = (6*300)/300(4+2) =1

Operations per byte = ¼

Exercise

Exercise 2: What is the operational intensity (Ops/byte) of the following code segment, assuming each data is 4 bytes long?

```
Code segment 2
For (i=1; i<99; i++){
     For (j=1; j<99; j++){
     I[i,j] = I[i-1, j-1]*K[0,0]+I[i-1,j]*K[0,1]+I[i-1,j+1]*K[0,2]+
            I[i, j-1]*K[1,0]+I[i,j]*K[1,1]+I[i,j+1]*K[1,2]+
            I[i+1, j-1]*K[2,0]+I[i+1,j]*K[2,1]+I[i+1,j+1]*K[2,2]
     }
}
```

Additional information (probably not relevant to this exercise) about the code segment. This operation used in image processing is like mathematical convolution of an image matrix I using the kernel K, which is also called the convolution matrix or mask. Operations such as blurring, sharpening, embossing, and edge-detection are performed using different kernels. Note that this code segment does not contain the normalization in the convolution operation.

Answer:

Number of operations = 98*98*(9+8)

= Approx. (10,000)*17 = 170,000 operations

Number of data read = 100*100+9 (Matrices I and K)

Amount of data read = (10,000+9)*4 bytes

Number of data written = 100*100 (Matrix I)

Amount of data (bytes) written = (10,000*4) bytes

Total data written to or read from memory = Approx. (20,000*4) bytes = (80,000) bytes

Operations/byte = Approx. 2

FMA and MAC

Modern processors implement Fused Multiply Add (FMA) or Multiply and Accumulate (MAC) instructions. Given three operands a, b, and c, the FMA can compute any one of the following in one step;

$y = a+(b*c)$

$y = a-(b*c)$

$y = -a+(b*c)$

$y = -a-(b*c)$

Note: A processor with FMA can perform the (9+8) operations in Code segment 2 in 9 time steps.

Exercise

Exercise 3: What is the operational intensity (Ops/byte) of the following code segment, assuming each data is 4 bytes long?

```
Code segment 3
For (i=1; i<99; i++){
     For (j=1; j<99; j++){
     I[i,j] = I[i-1, j-1]+I[i-1,j]+I[i-1,j+1]+
           I[i, j-1]+I[i,j]+I[i,j+1]+
           I[i+1, j-1]+I[i+1,j]+I[i+1,j+1]
     }
}
```

Exercise

Exercise 3: What is the operational intensity (Ops/byte) of the following code segment, assuming each data is 4 bytes long?

```
Code segment 3
For (i=1; i<99; i++){
     For (j=1; j<99; j++){
     I[i,j] = I[i-1, j-1]+I[i-1,j]+I[i-1,j+1]+
            I[i, j-1]+I[i,j]+I[i,j+1]+
            I[i+1, j-1]+I[i+1,j]+I[i+1,j+1]
     }
}
```

Answer: 1 operation/byte.

Exercise

Note:
Assuming there is one functional unit to perform addition, the number of time steps to complete the operations in each iteration is 8, which is comparable to (about the same as) what we achieved with an FMA and Code segment 2 that had almost two times the operations in Code segment 1.

Observation: Code segment 3 does not utilize FMA.

Exercise

```
Code segment 1
For (i=0; i<300; i++){
    C_re[i] = A_re[i]*B_re[i] – A_im[i]*B_im[i];
    C_im[i] = A_re[i]*B_im[i] + A_im[i]*B_re[i];
}
```

Assuming that

- there are 10 FMAs in a processor
- Code segment 1 is rewritten with loop unrolling to utilize the 10 FMAs in SIMD mode.

Question: How much speed-up can you expect while executing Code segment 2?

Plotting Rooflines

Given the following machine specifications for a core:

(a) Peak performance with SIMD of 8 operations/cycle

(b) Scalar Peak performance (without SIMD) 2 ops/cycle

(c) Maximal memory bandwidth = 1 double/cycle = 8 bytes/cycle

(d) Maximal bandwidth achievable without spatial locality = (1/8) double/cycle = 1 byte/cycle,

Draw a roofline plot for double precision floating point operations on the core. The units for x-axis and y-axis are ops/byte and ops/cycle, respectively.

Note: (1) that the bandwidths always refer to off-chip data transfers, and (2) the plots need to be log-log plots.

Plotting Rooflines

Specifically, the plot should contain 4 lines:

(a) Upper bound based on peak performance with SIMD

(b) Upper bound based on scalar peak performance

(c) Upper bound based on the maximal memory bandwidth

(d) Upper bound based on the maximal bandwidth achievable without spatial locality (e.g., random access of doubles in a very large array).

Finally answer the following: What is the minimal operational intensity that a non-vectorizable (cannot use SIMD operation) computation without spatial locality needs to have to be compute bound?

Roofline: Drawing Bounds Based on Memory Bandwidth

Given: x = operations/byte = operational intensity (OI) ; and

y= operations/cycle; What is the relationship between x, the memory bandwidth (B) in bytes/sec, and the upper bound on y, the number of operations/cycle?

Number of operations/cycle <= (memory bandwidth*operational intensity);

Thus, the bounding line is given by y = B*x , a straight line with slope = B and y-intercept = 0.

Since we need a log-log plot taking logarithms on both sides;

log(y) = log(B) + log(x) – fortunately the straight line y = 8*x translated to a straight line in the log-log scale with slope =1 and with log(y) = log(B) for x=1.

Question: Why do we need log-log plots?

References

- Presentation by Sam Williams at https://ftg.lbl.gov/assets/staff/swwilliams/talks/parlab08roofline.pdf. We use this to illustrate how this model is used to analyze the performance of various kernels executing on a few multicore processors.
- Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52, 4 (April 2009), 65-76. DOI=10.1145/1498765.1498785 http://doi.acm.org/10.1145/1498765.1498785