The Roofline Model

High Performance Computing for Science and Engineering

CSE**lab**
Computational Science & Engineering Laboratory
http://www.cse-lab.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Additional Literature

- Samuel Williams, Andrew Waterman, David Patterson
  “An Insightful Visual Performance Model for Multicore Architectures”

- Samuel Williams, David Patterson
  “The Roofline Model: a pedagogical tool for program analysis and optimization” (slides)

Motivation

- Performance is not intuitive
- Goals:
  - Graphical aid for realistic expectations of performance
  - Show hardware limitations for a given code
  - Prioritize optimizations

Assumptions and Limitations

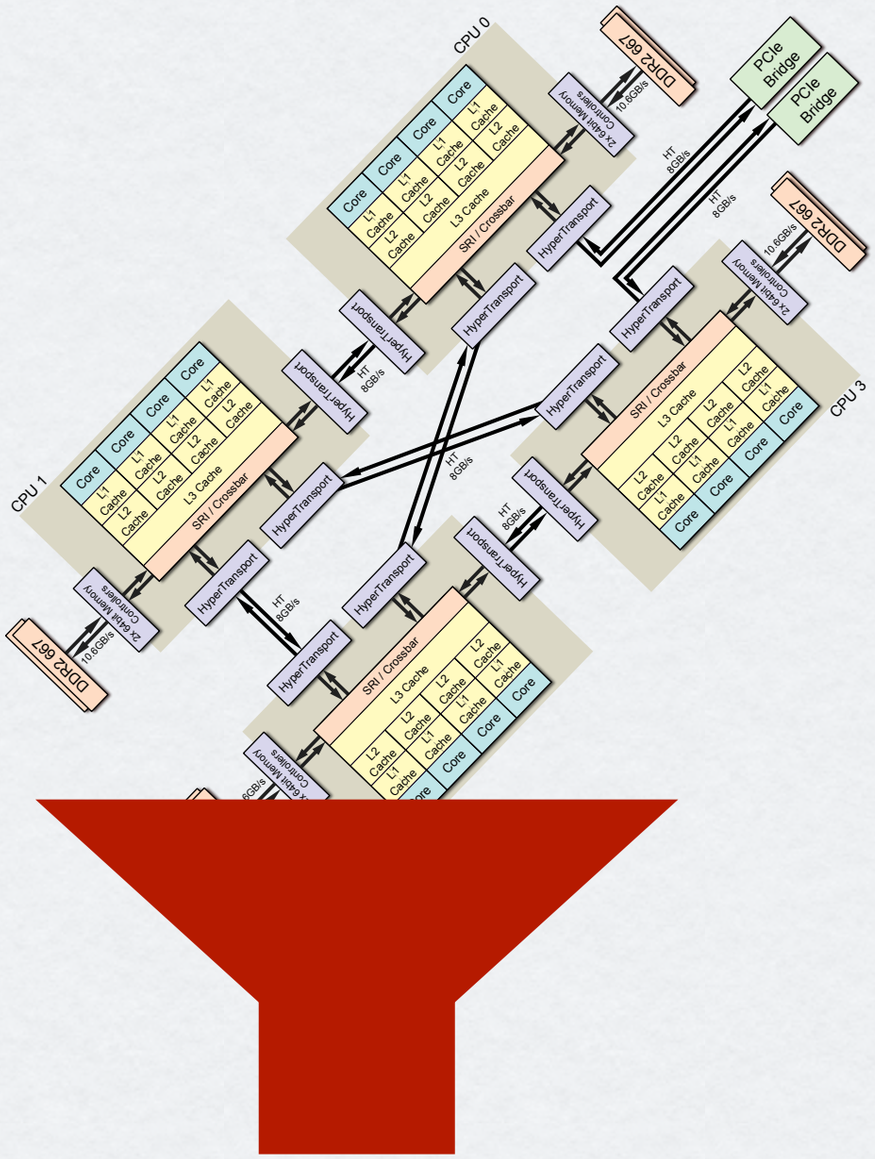
Main assumption:

- Transfer-computation overlap

Limitations:

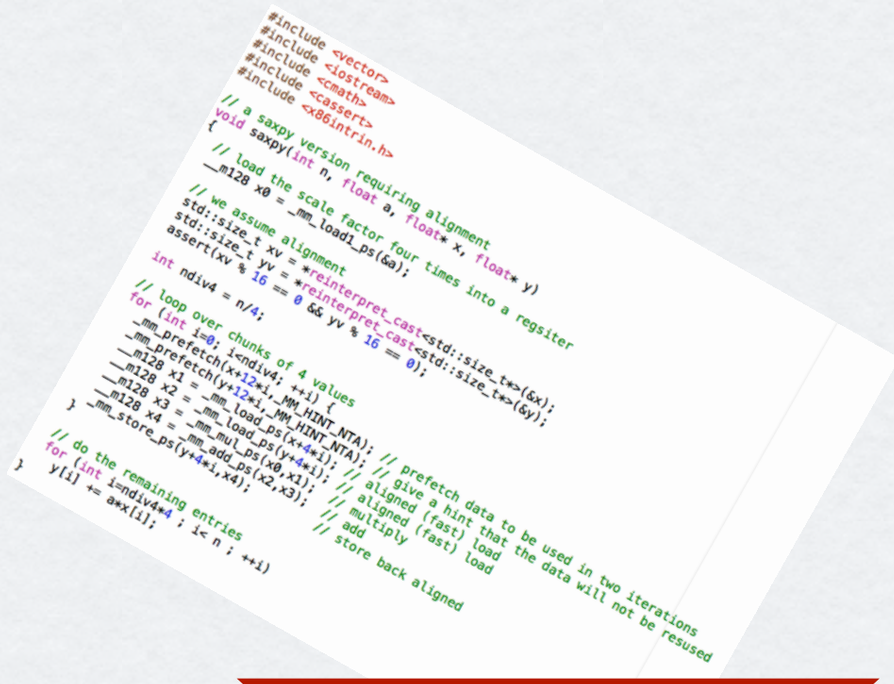
- Visual instrument, not exact
  - Making it a precise instrument is complicated...
- How to handle bandwidth in a network?

Abstraction



GFLOP/s

GB/s



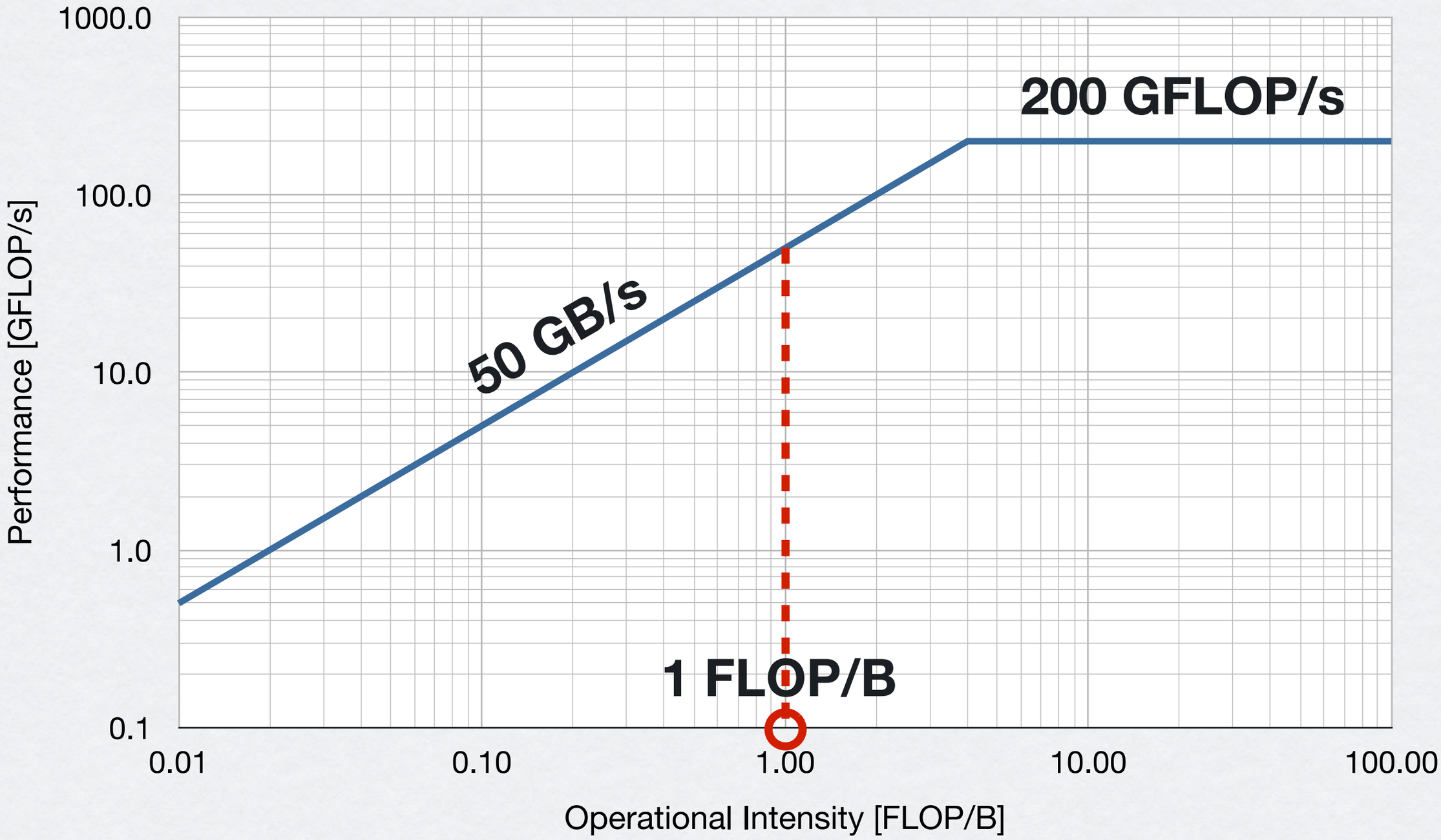
```
#include <vector>
#include <iostream>
#include <cmath>
#include <cassert>
#include <x86intrin.h>
// a saxpy version requiring alignment
void saxpy(int n, float a, float* x, float* y)
{
  // load the scale factor four times into a regsiter
  __m128 x0 = _mm_load1_ps(&a);
  // we assume alignment
  std::size_t xv = *reinterpret_cast<std::size_t*>(&x);
  std::size_t yv = *reinterpret_cast<std::size_t*>(&y);
  assert(xv % 16 == 0 && yv % 16 == 0);
  int ndiv4 = n/4;
  // loop over chunks of 4 values
  for (int i=0; i<ndiv4; ++i) {
    _mm_prefetch(x+12*i, _MM_HINT_NTA); // prefetch data to be used in two iterations
    _mm_prefetch(y+12*i, _MM_HINT_NTA); // give a hint that the data will not be resused
    __m128 x1 = _mm_load_ps(x+4*i); // aligned (fast) load
    __m128 x2 = _mm_load_ps(y+4*i); // aligned (fast) load
    __m128 x3 = _mm_mul_ps(x0,x1); // multiply
    __m128 x4 = _mm_add_ps(x2,x3); // add
    _mm_store_ps(y+4*i,x4); // store back aligned
  }
  // do the remaining entries
  for (int i=ndiv4*4 ; i< n ; ++i)
    y[i] += a*x[i];
}
```

FLOP/B

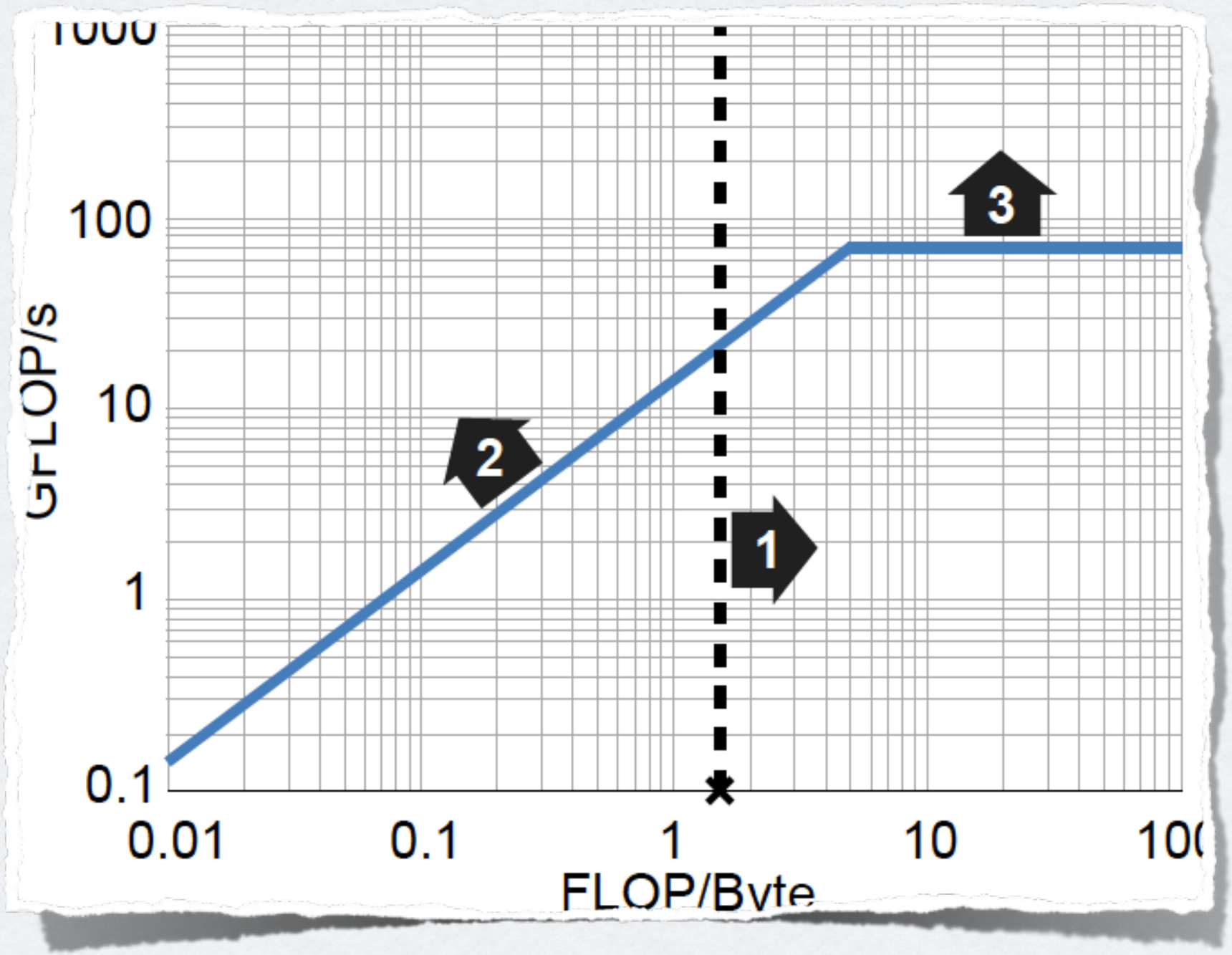
Operational Intensity:

FLOP-to-byte of off-chip memory transfers

Roofline


1000.0
100.0
10.0
1.0
0.1
Performance [GFLOP/s]
200 GFLOP/s
50 GB/s
1 FLOP/B
0.01
0.10
1.00
10.00
100.00
Operational Intensity [FLOP/B]

Optimization



1. Locality
2. Communication
3. Computation

Operational Intensity

Given

```
for (int ix=1; ix<N-1; ix++)
  out(ix) = in(ix-1)-2*in(ix)+in(ix+1)
```

where `in` and `out` are float arrays of size `N`

1. What is the amount of floating point operations?

2. What is the amount of memory accesses from main memory if:

   a) there is no caching

   b) there is a perfect cache of infinite size

Operational Intensity

```
for (int ix=1; ix<N-1; ix++)
  out(ix) = in(ix-1)-2.*in(ix)+in(ix+1)
```

Floating point operations: **3***(N-2) FLOP

Memory accesses (no caching): **4***(N-2) floats accessed

every data accessed is counted

Memory accesses (perfect caching): **2***N floats accessed

data is read only once and written only once

High Performance Computing for
Computational Science and Engineering

The Roofline Model

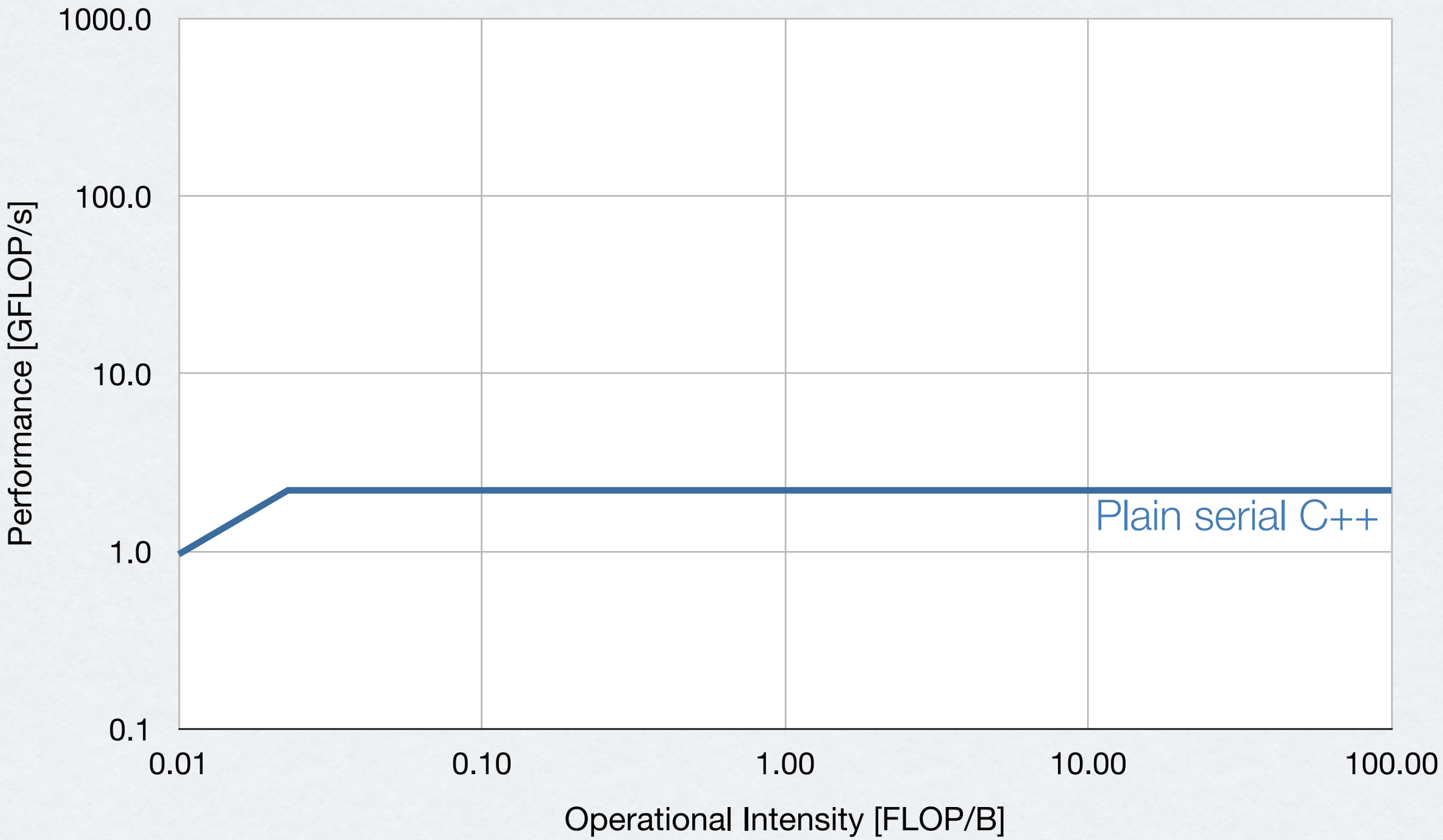
Roofline-Based Optimization of a 2D Diffusion Stencil

November 20th, 2013

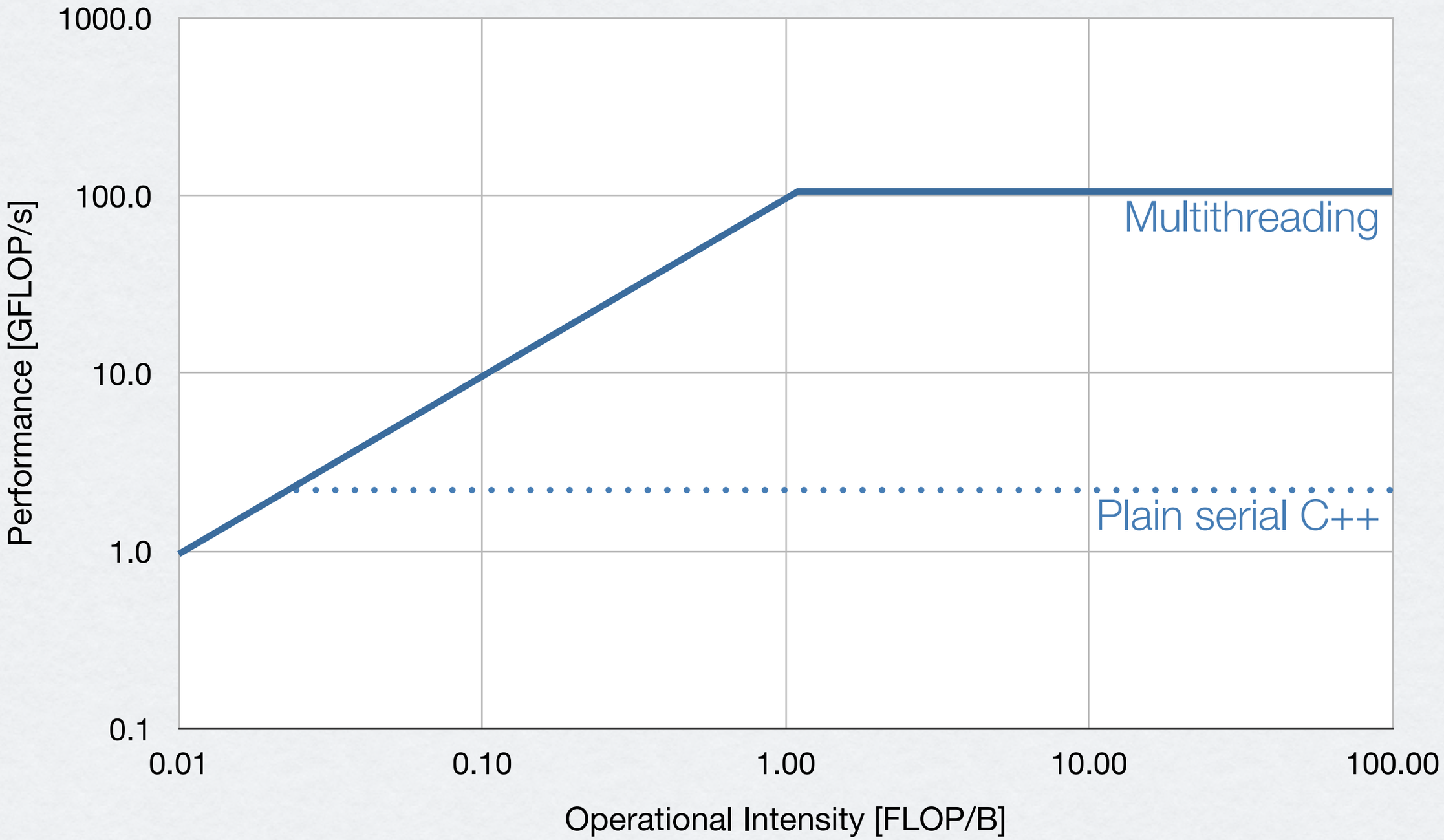
CSE**lab**
Computational Science & Engineering Laboratory
http://www.cse-lab.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

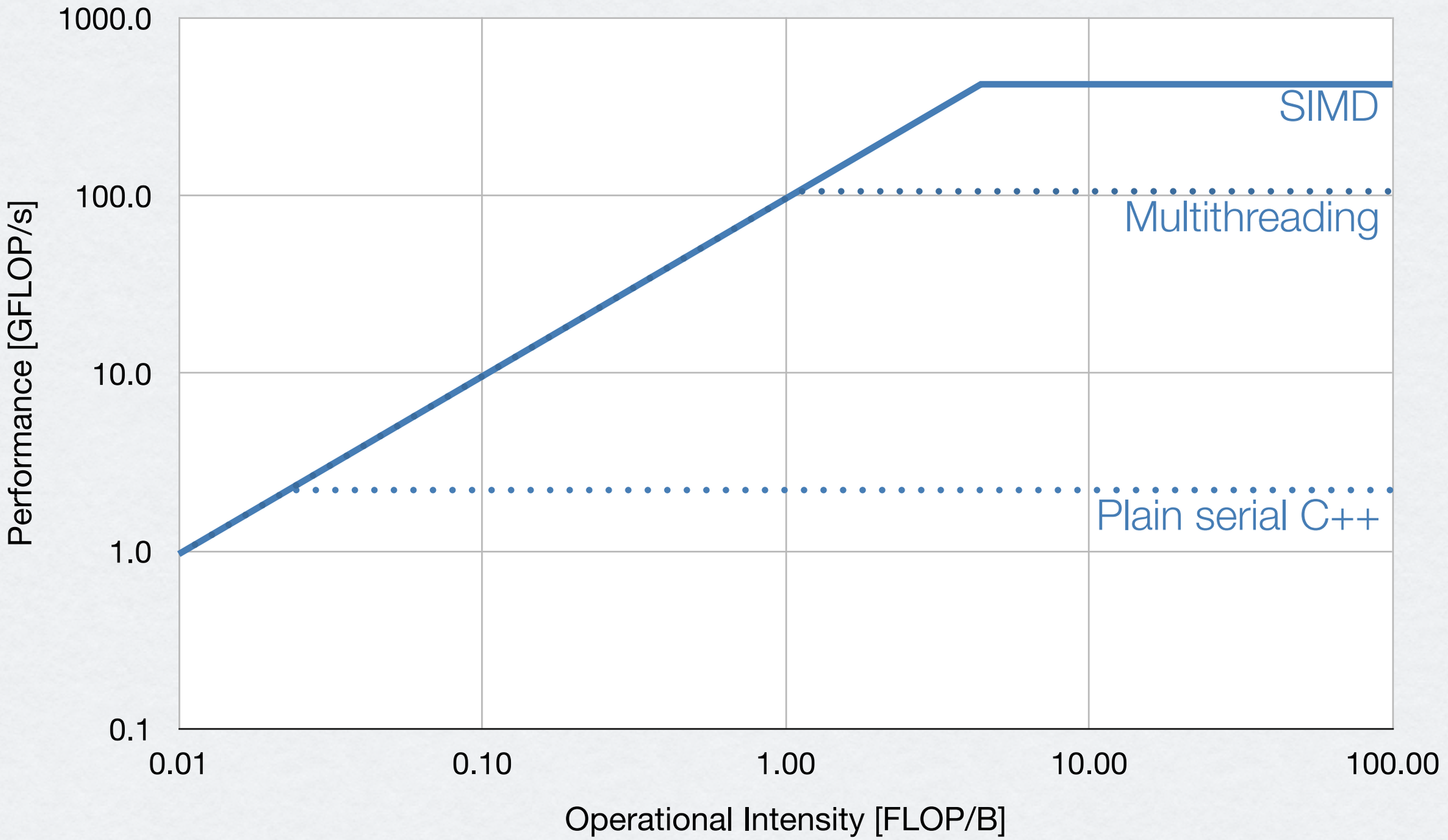
Ceilings on Brutus (single precision)



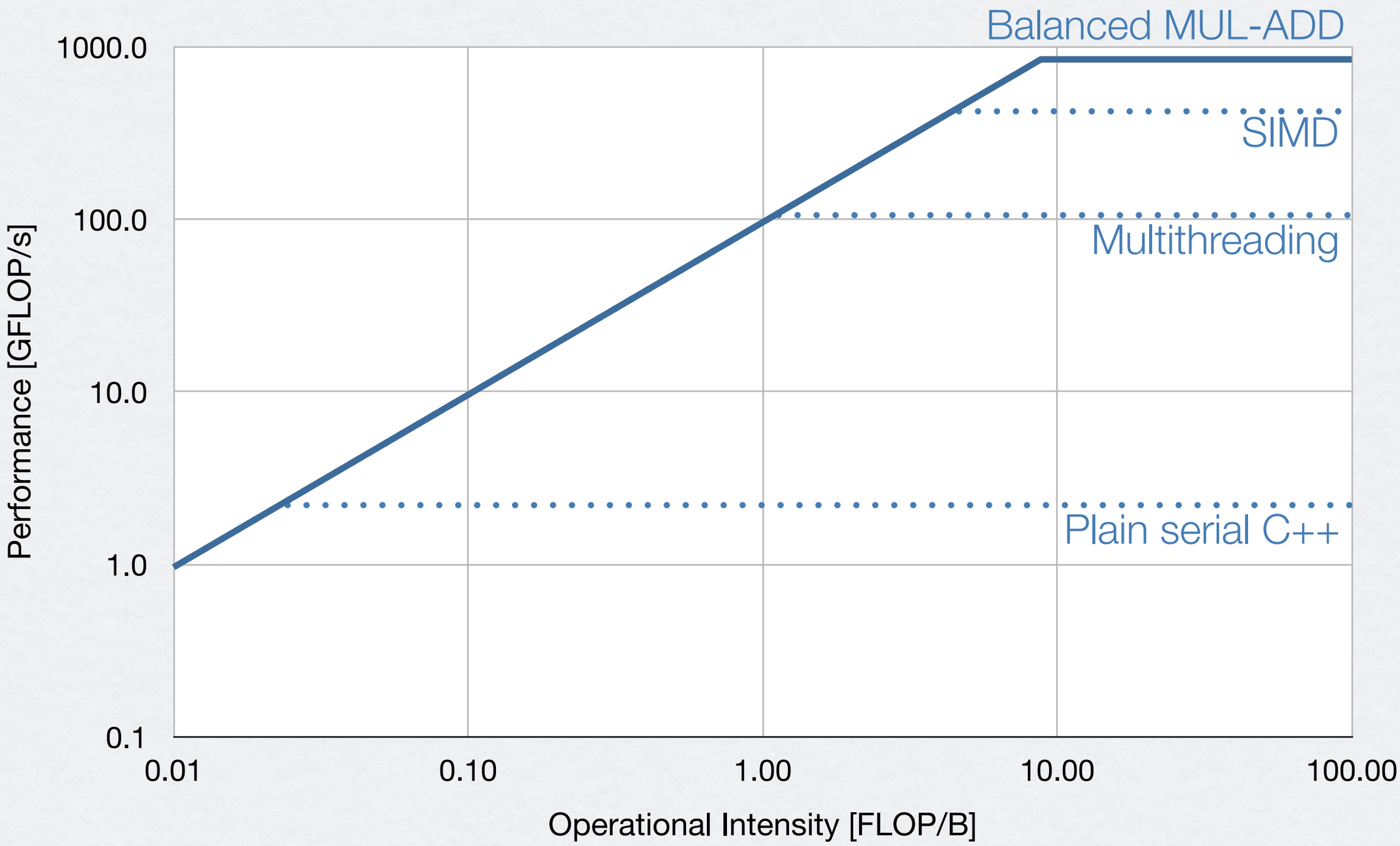
Ceilings on Brutus (single precision)



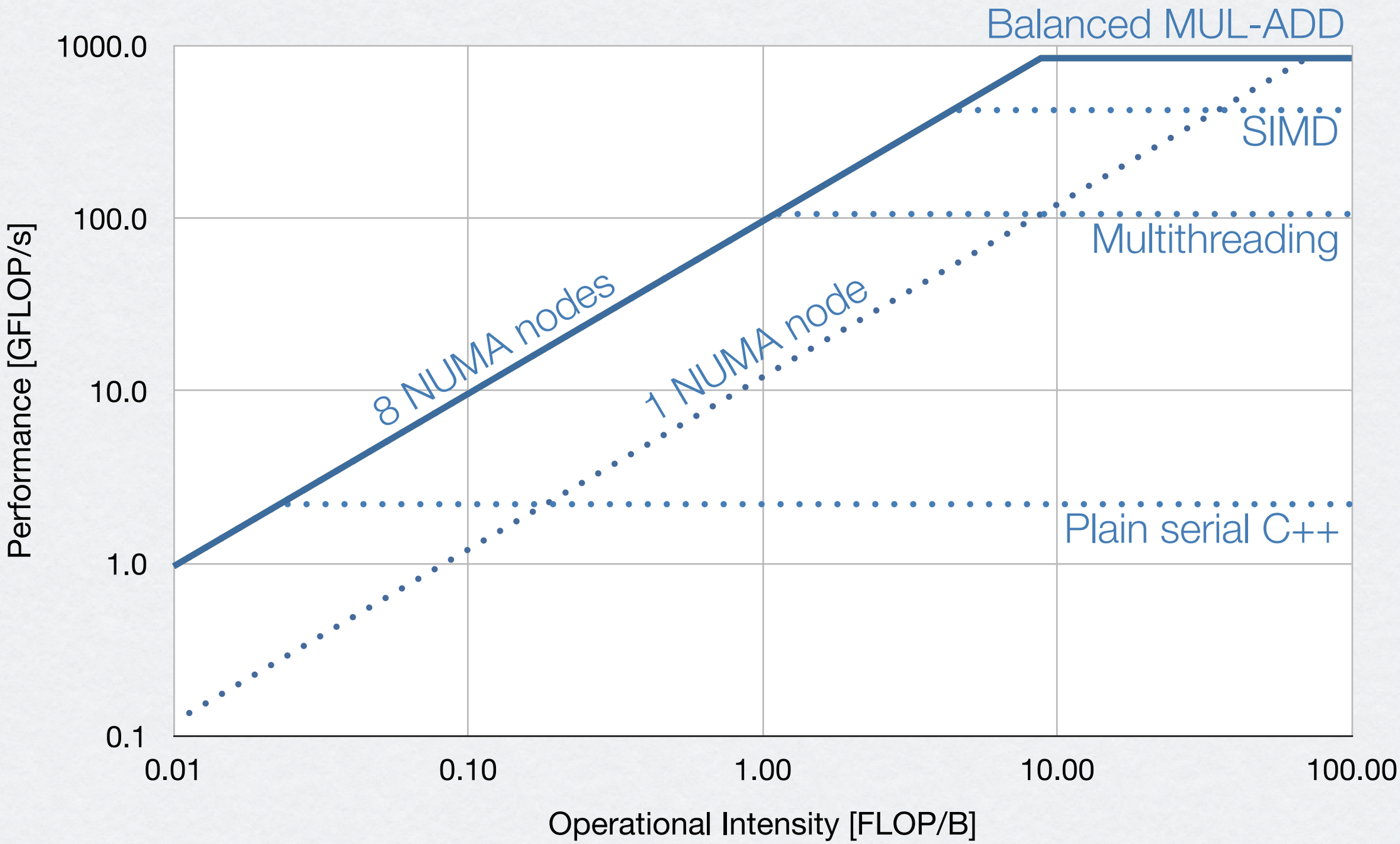
Ceilings on Brutus (single precision)



Ceilings on Brutus (single precision)



Ceilings on Brutus (single precision)



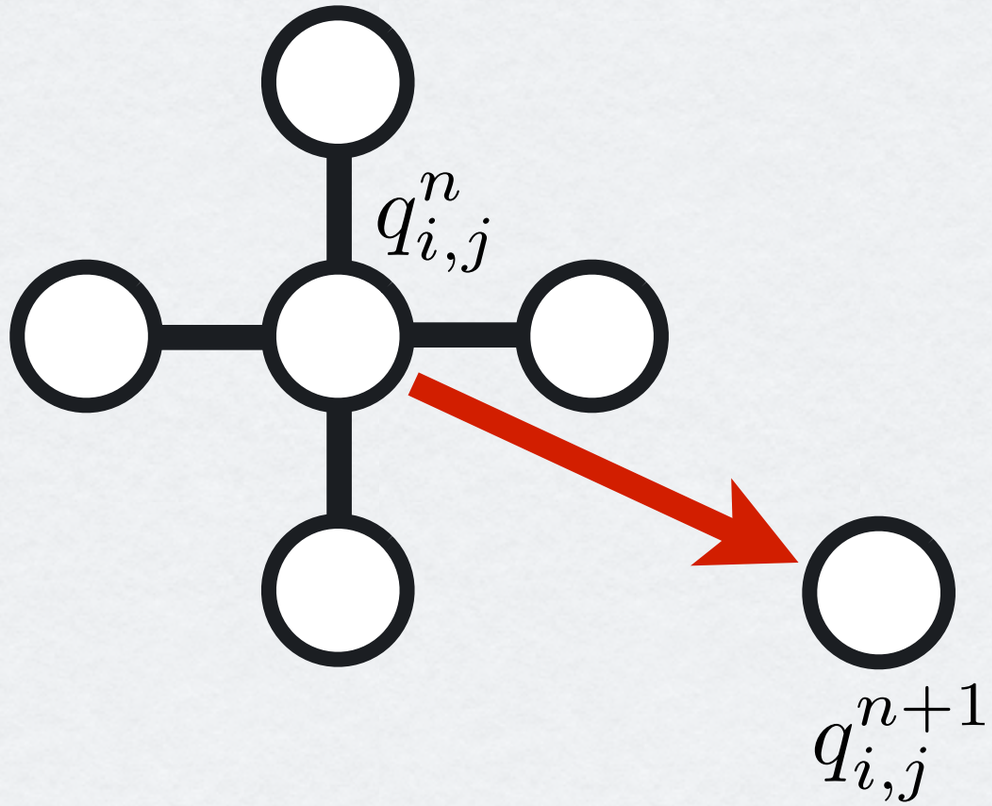
2D Heat Equation

2D heat equation:

$$\frac{\partial q}{\partial t} - D\Delta q = 0$$

q: singe precision (4 Bytes)

5 point stencil



Algorithm:

1. Laplace Operator

$$RHS_{i,j} = C_1(q^n_{i+1,j} + q^n_{i-1,j} + q^n_{i,j+1} + q^n_{i,j-1} - 4q^n_{i,j})$$

2. Forward Euler Operator

$$q^{n+1}_{i,j} = q^n_{i,j} + \delta t \cdot RHS_{i,j}$$

A-Priori Performance Analysis

$$RHS_{i,j} = C_1(q^n_{i+1,j} + q^n_{i-1,j} + q^n_{i,j+1} + q^n_{i,j-1} - 4q^n_{i,j})$$

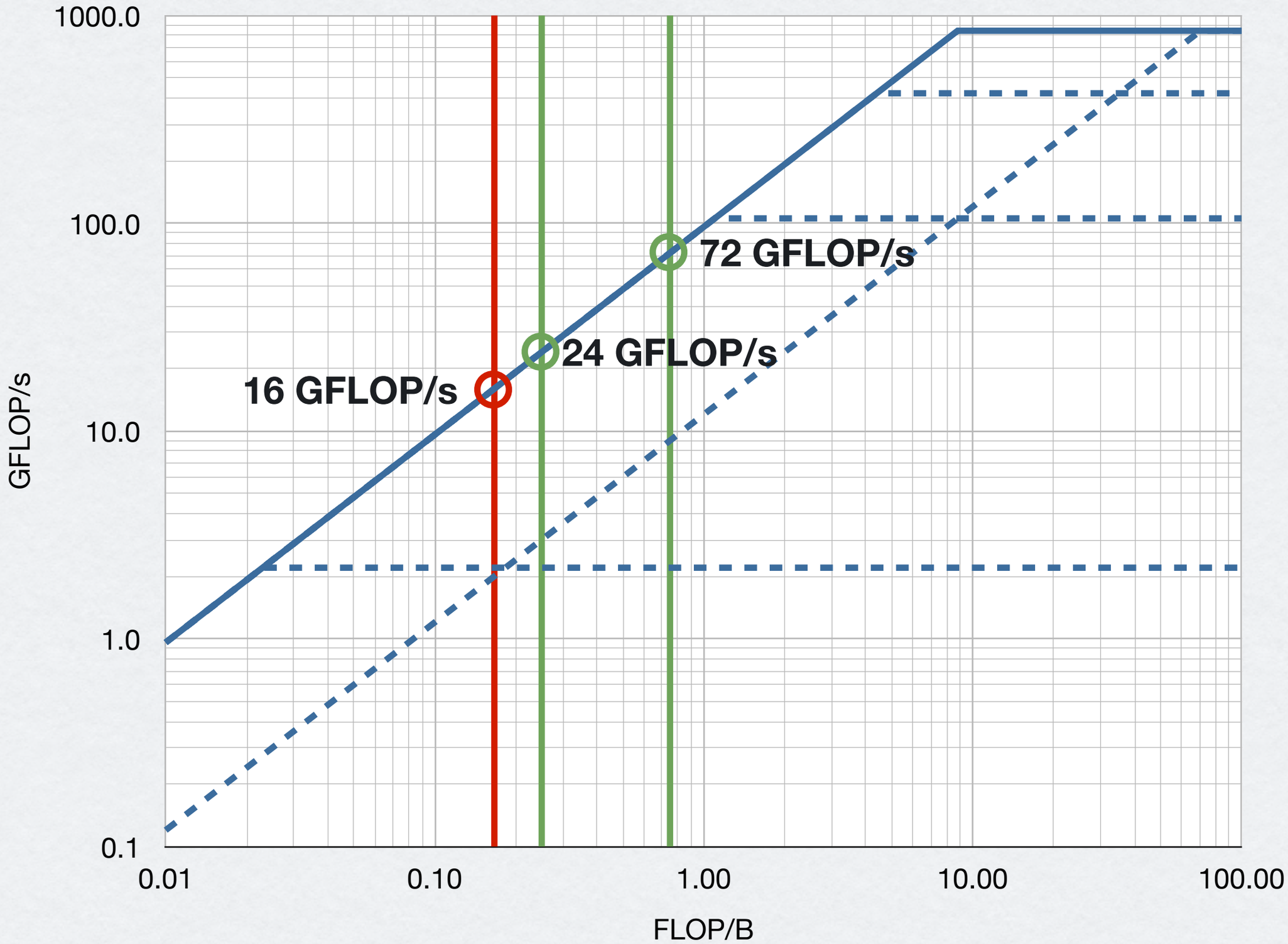
- Floating point operations per point: 4 ADD + 2 MUL
- Memory accesses per point:
  - Worst case: 5 read + 1 write
  - Best case: 1 read + 1 write
- Operational Intensity:
  - Worst case: 6 FLOP / (6*4 B) = 0.25 FLOP/B
  - Best case: 6 FLOP / (2*4 B) = 0.75 FLOP/B

A-Priori Performance Analysis

$$q_{i,j}^{n+1} = q_{i,j}^{n} + \delta t \cdot RHS_{i,j}$$

- Floating point operations per point: 1 ADD + 1 MUL
- Memory accesses per point:
  - Worst case: 2 read + 1 write
  - Best case: 2 read + 1 write
- Operational Intensity:
  - Worst case: 2 FLOP / (3*4 B) = 0.17 FLOP/B
  - Best case: 2 FLOP / (3*4 B) = 0.17 FLOP/B

Roofline


1000.0
100.0
10.0
1.0
0.1
GFLOP/s
0.01
0.10
1.00
10.00
100.00
FLOP/B
72 GFLOP/s
24 GFLOP/s
16 GFLOP/s

Algorithm v2

$$RHS_{i,j} = C_1(q^n_{i+1,j} + q^n_{i-1,j} + q^n_{i,j+1} + q^n_{i,j-1} - 4q^n_{i,j})$$

$$q^{n+1}_{i,j} = C_0 q^n_{i,j} + C^*_1(q^n_{i+1,j} + q^n_{i-1,j} + q^n_{i,j+1} + q^n_{i,j-1})$$

$$q^{n+1}_{i,j} = q^n_{i,j} + \delta t \cdot RHS_{i,j}$$

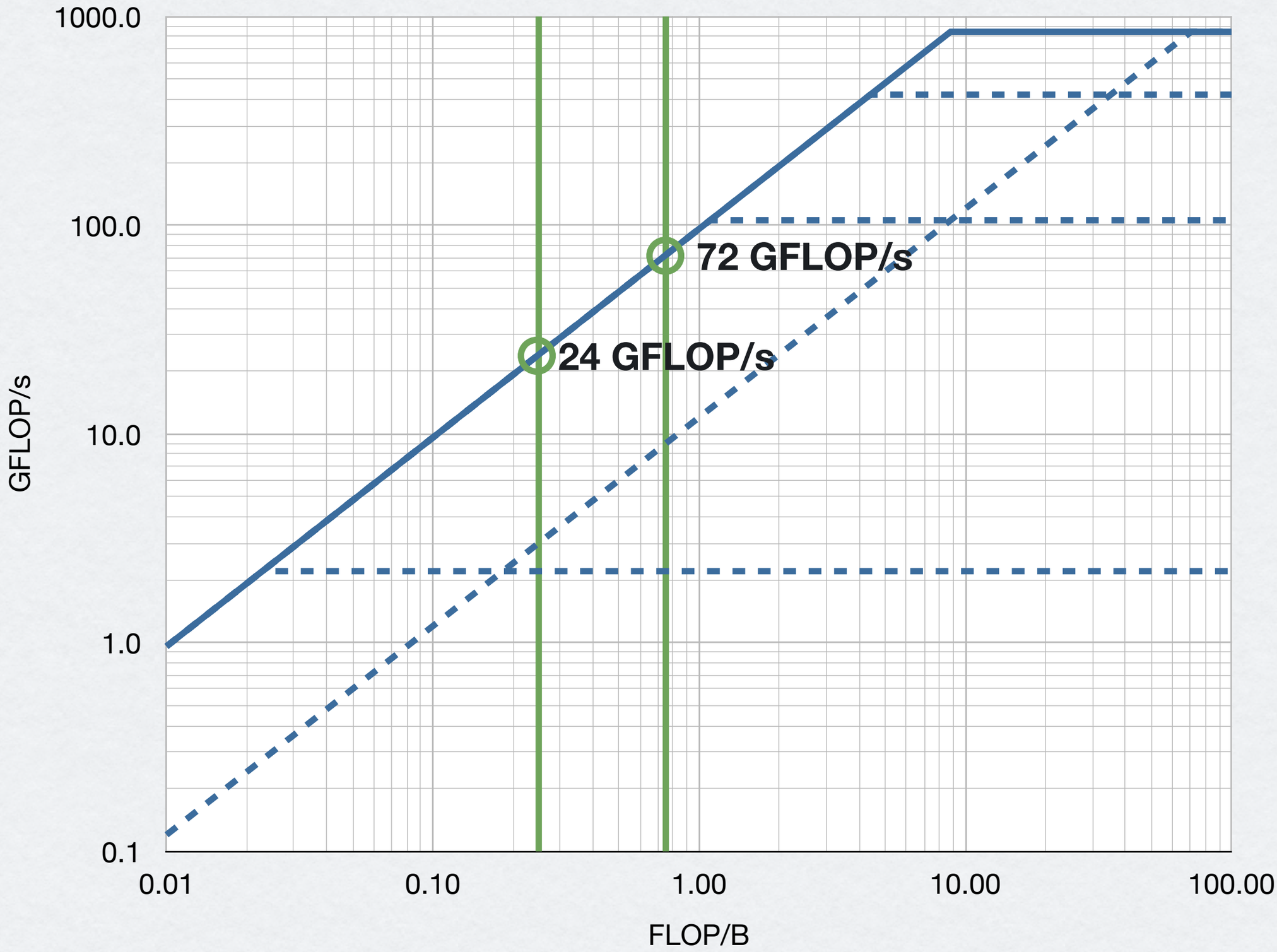
A-Priori Performance Analysis

$$q_{i,j}^{n+1} = C_0 q_{i,j}^n + C_1^* (q_{i+1,j}^n + q_{i-1,j}^n + q_{i,j+1}^n + q_{i,j-1}^n)$$

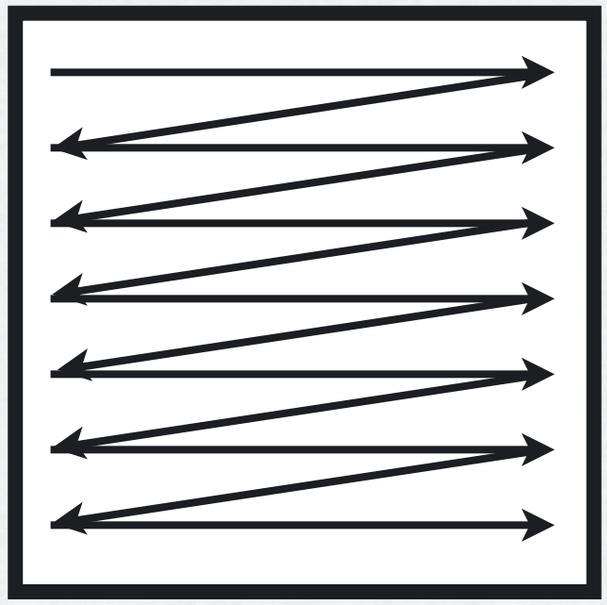
- Floating point operations per point: 4 ADD + 2 MUL
- Memory accesses per point:
  - Worst case: 5 read + 1 write
  - Best case: 1 read + 1 write
- Operational Intensity:
  - Worst case: 6 FLOP / (6*4 B) = 0.25 FLOP/B
  - Best case: 6 FLOP / (2*4 B) = 0.75 FLOP/B

!

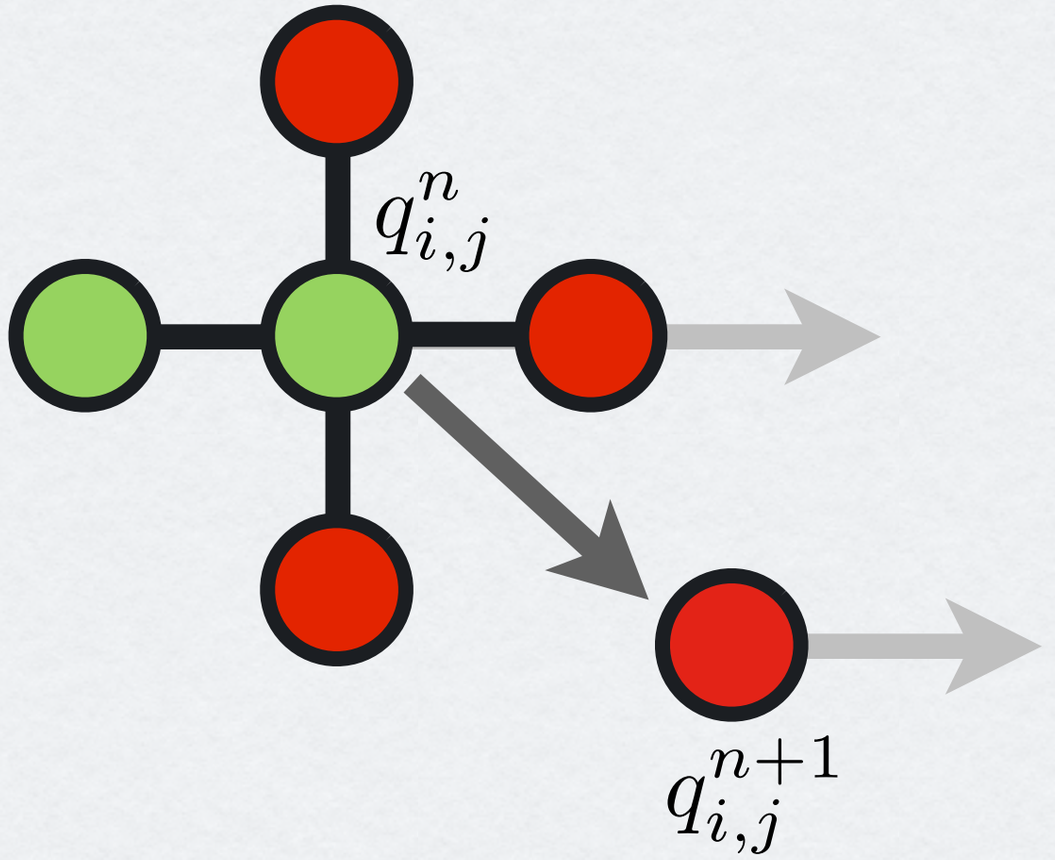
Roofline


1000.0
100.0
10.0
1.0
0.1
GFLOP/s
72 GFLOP/s
24 GFLOP/s
0.01
0.10
1.00
10.00
100.00
FLOP/B

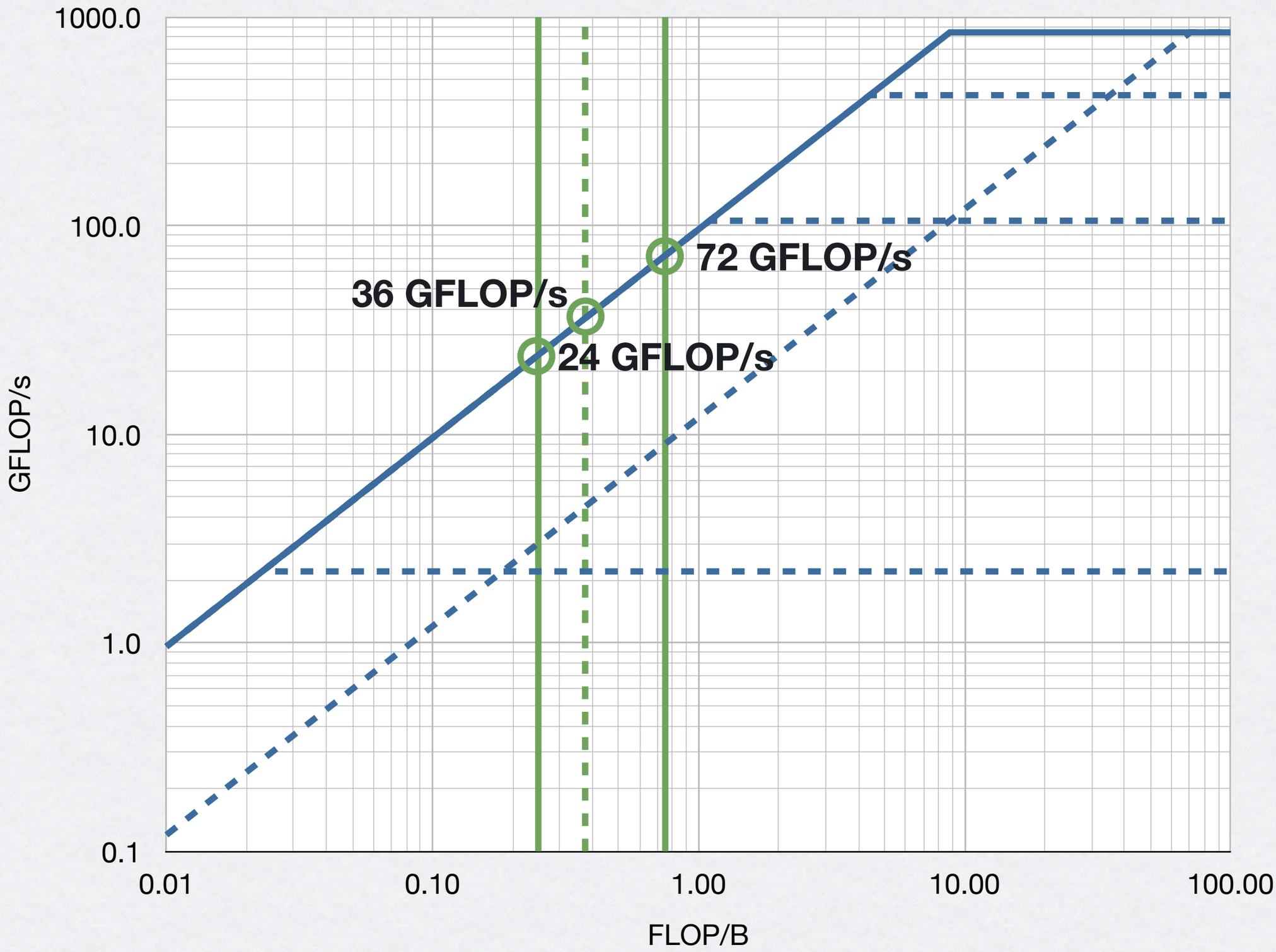
A More Accurate Analysis



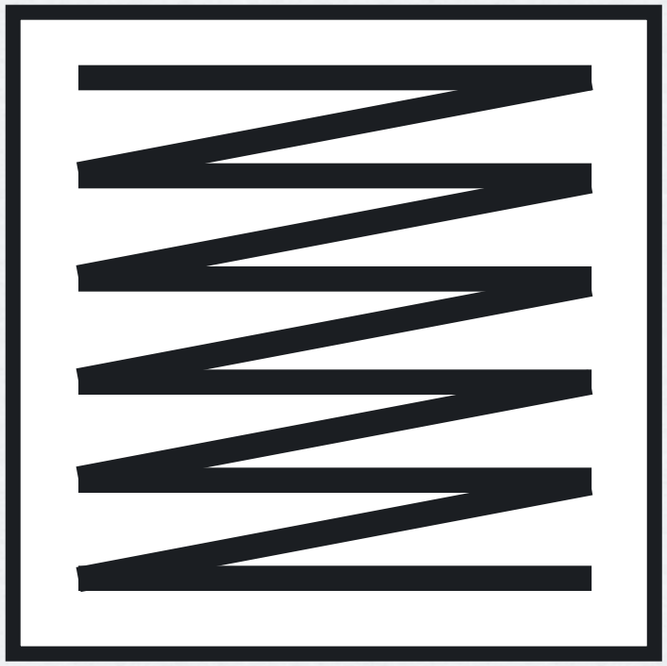
- We have locality!
- Memory accesses per point:
  - 3 read + 1 write
- Operational Intensity:
  - 6 FLOP / (4*4 B) = 0.375 FLOP/B



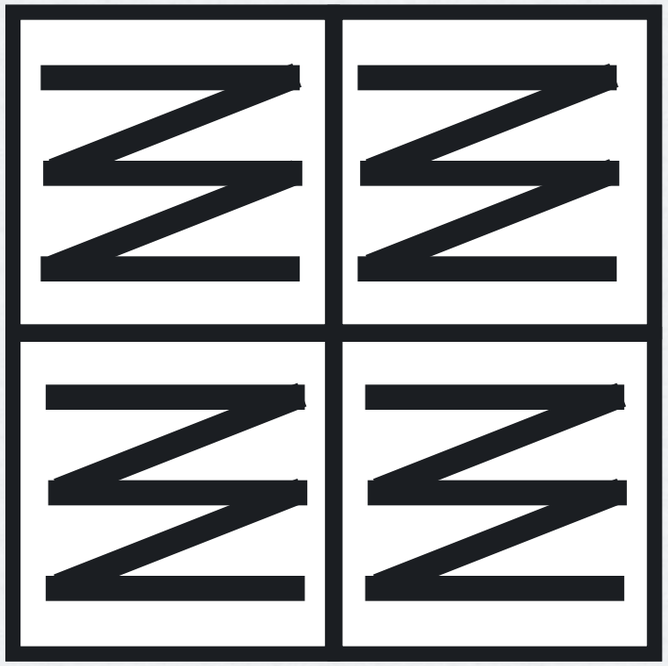
Roofline


1000.0
100.0
10.0
1.0
0.1
GFLOP/s
0.01
0.10
1.00
10.00
100.00
FLOP/B
72 GFLOP/s
36 GFLOP/s
24 GFLOP/s

Improving Locality

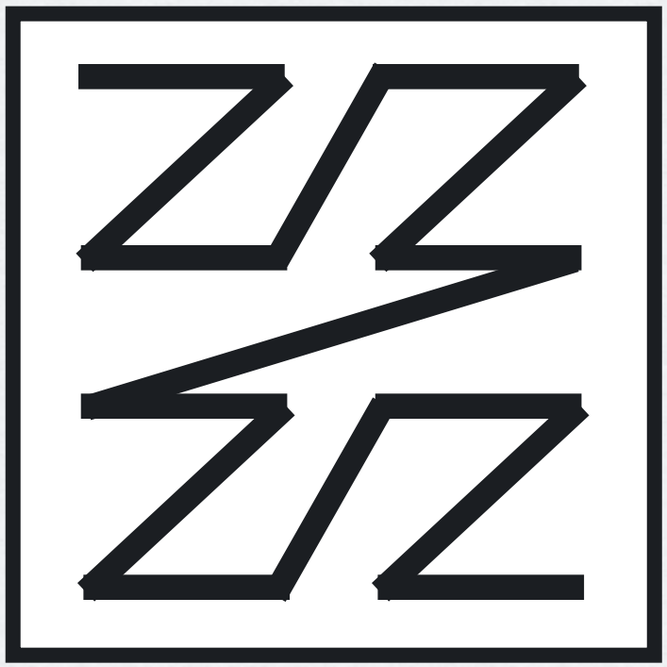


Linear

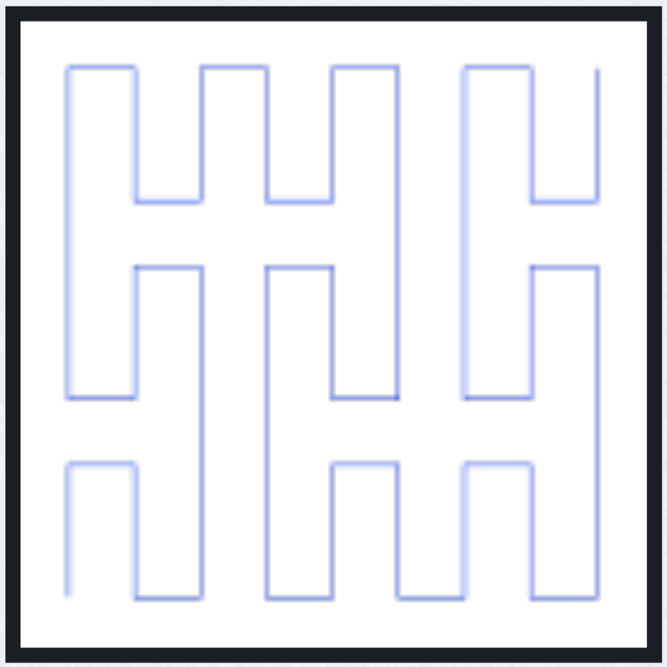


Blocked

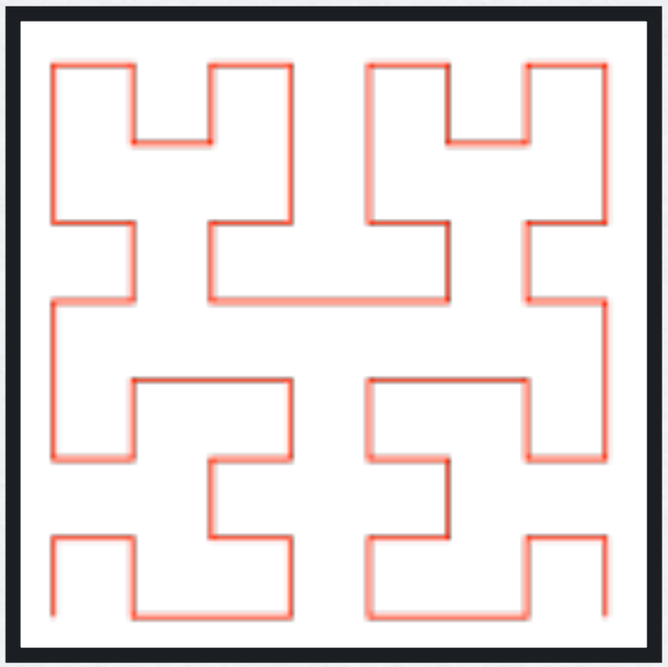
Blocking Hierarchy



Morton or Z-Order



Peano



Hilbert