# Should I port my code to a GPU?

Casey Battaglino - Aparna Chandramowlishwaran - Jee Choi -
Kent Czechowski - M. Efe Guney (*UC Davis*) - Chris McClanahan -
Logan Moon - Dave Noble - Aashay Shringarpure (*Google*) -
**Richard (Rich) Vuduc**

**Clusters, Clouds, and Grids for Scientific Computing**
Flat Rock, North Carolina - September 9, 2010

Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage

## A: It depends. Opportunity cost?

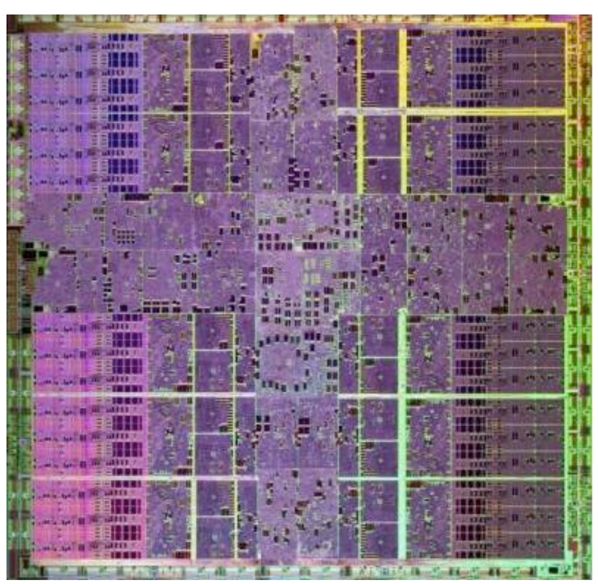
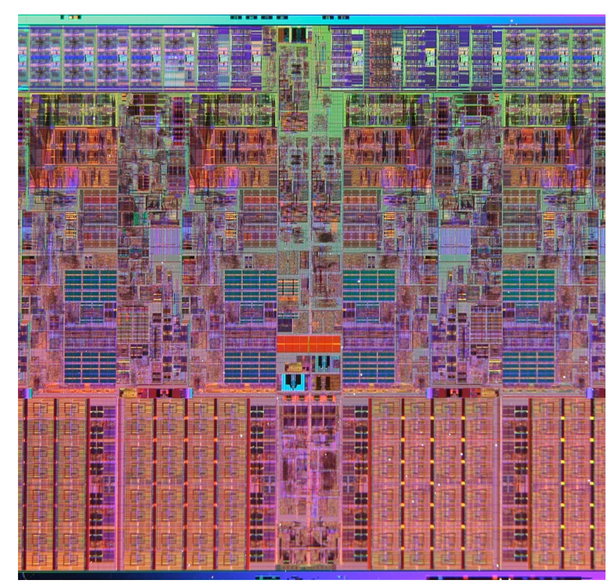
(1) Who are you?
(2) What is your app?
(3) What are your performance, productivity, and portability goals?

For most of us in this room, I'd say, "yes."
For the "average" apps developer, I'd say "not yet."

# Q: Pay-off of a GPU port?

(Posed to me by Scott Klasky at ORNL)

- Meta-analysis, for **semi-irregular** sci. comp. + data analytics apps (sparse iterative + direct solvers; tree-based particle methods)
- **A: Given roughly same level of tuning & power*, …**

**GPU**  $\cong$  **x 2 CPUs**

*R. Vuduc, A. Chandramowlishwaran, J. Choi, M. E. Guney, A. Shringarpure. USENIX HotPar'10, June 2010.*

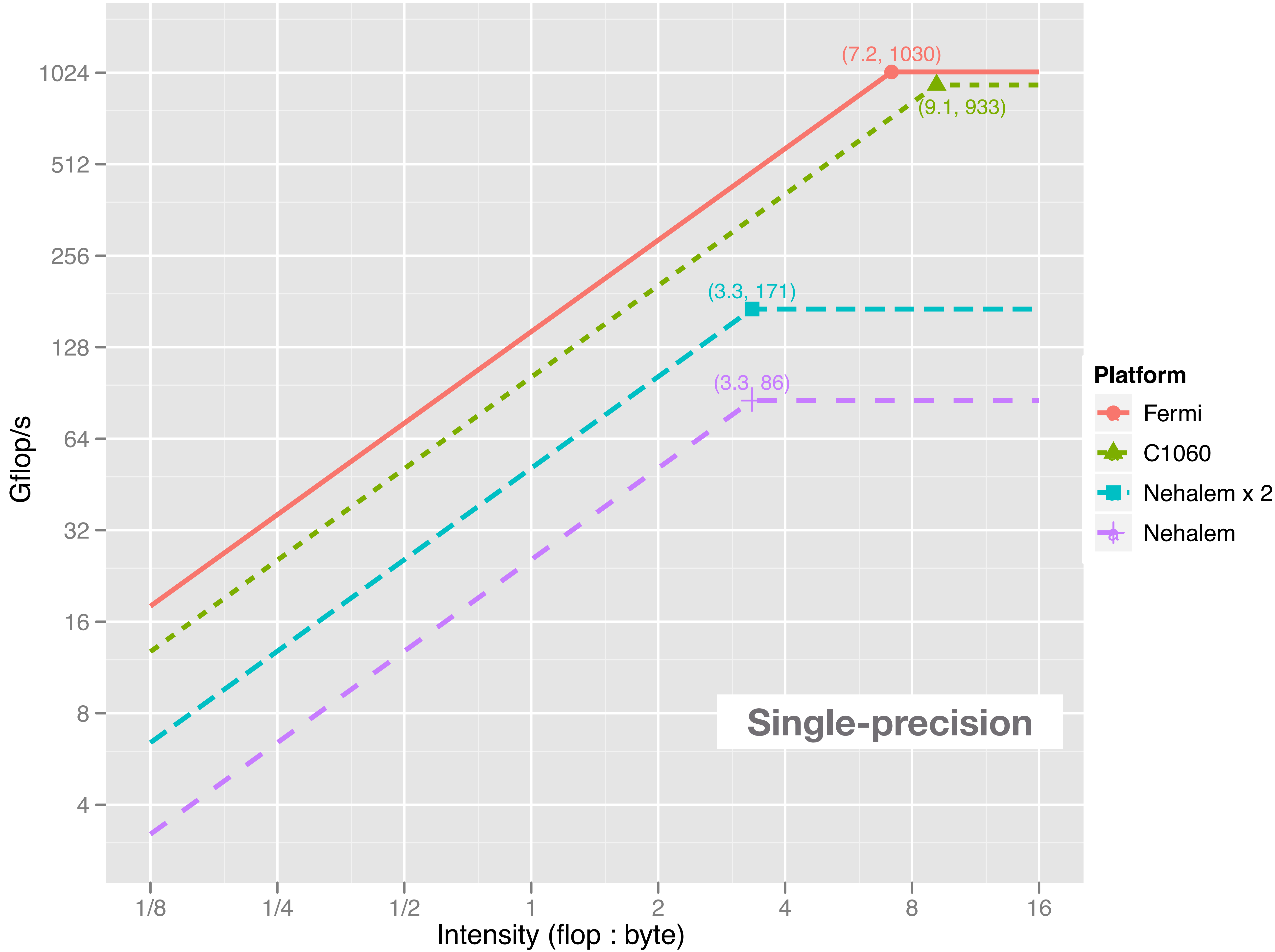
*Reason 1:* **The potential is real, but might be less than you expect.**

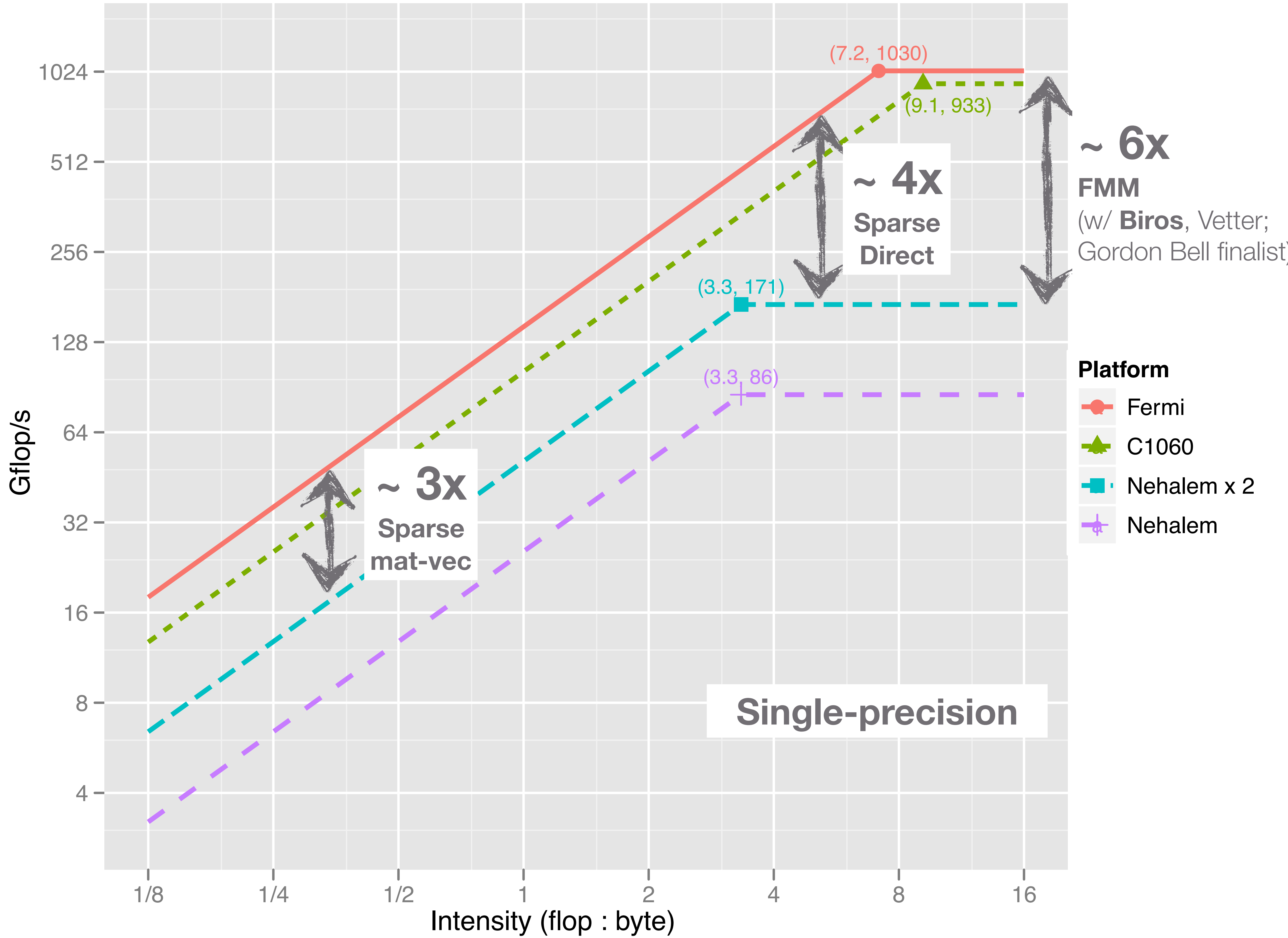
Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage

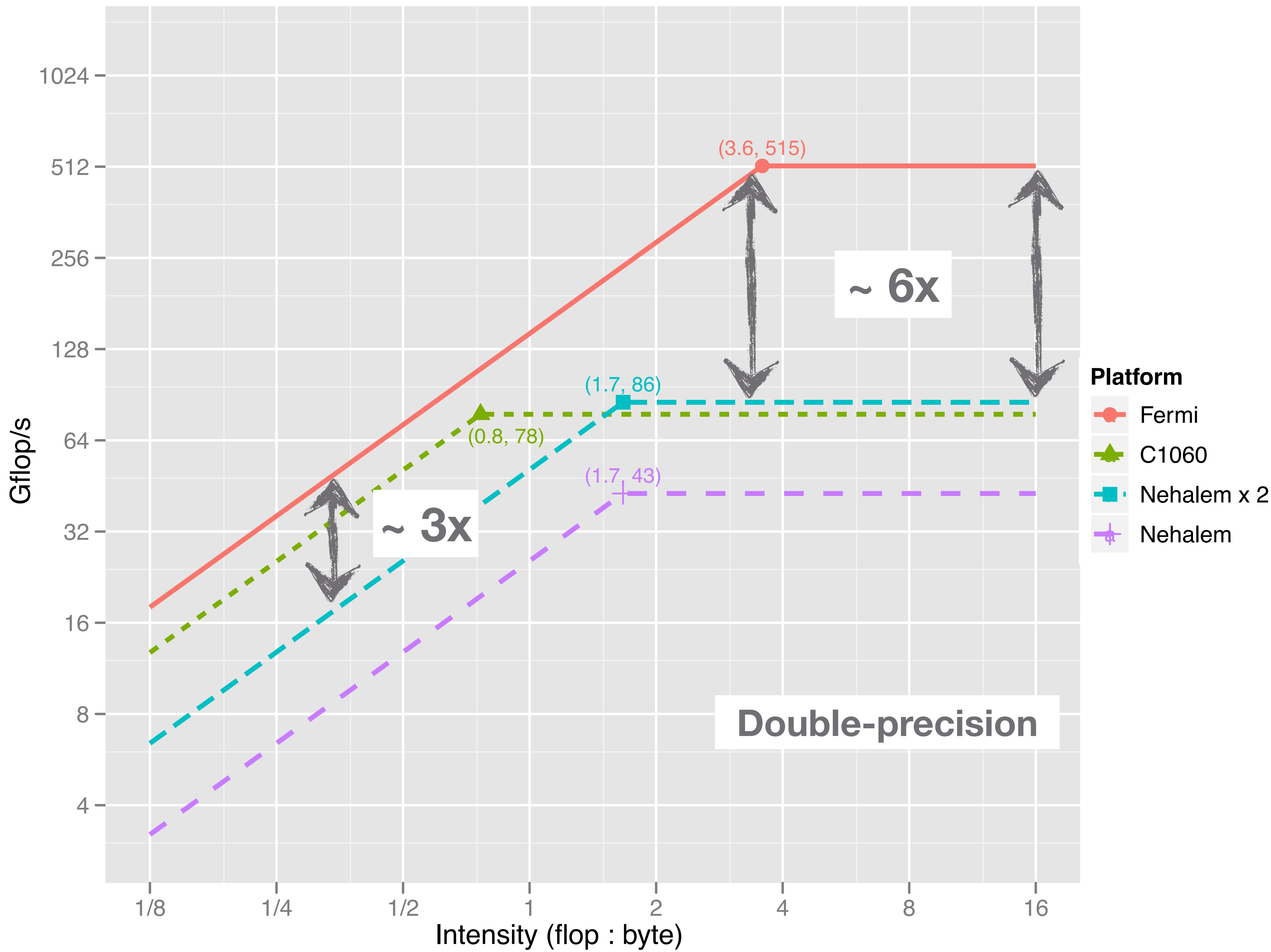
| Architecture | Intel Nehalem X5550 | NVIDIA T10P C1060 | NVIDIA GT200 GTX 285 | NVIDIA Fermi C2050 |
|---|---|---|---|---|
| GHz | 2.66 | 1.44 | 1.47 | 1.15 |
| Sockets | 2 | 1 | 1 | 1 |
| Cores per socket | 4 | 30 | 30 | 15* |
| Peak Gflop/s single (*double*) | 170.6 (*85.3*) | 933 (*78*) | 1060 (*88.4*) | 1030 (*515*) |
| Peak GB/s | 51.2 | 102 | 159 | 144 |
| **Sys. Watts (sockets only)** | **375 (200)** | **200** | **204** | **247** |

**+ host?**

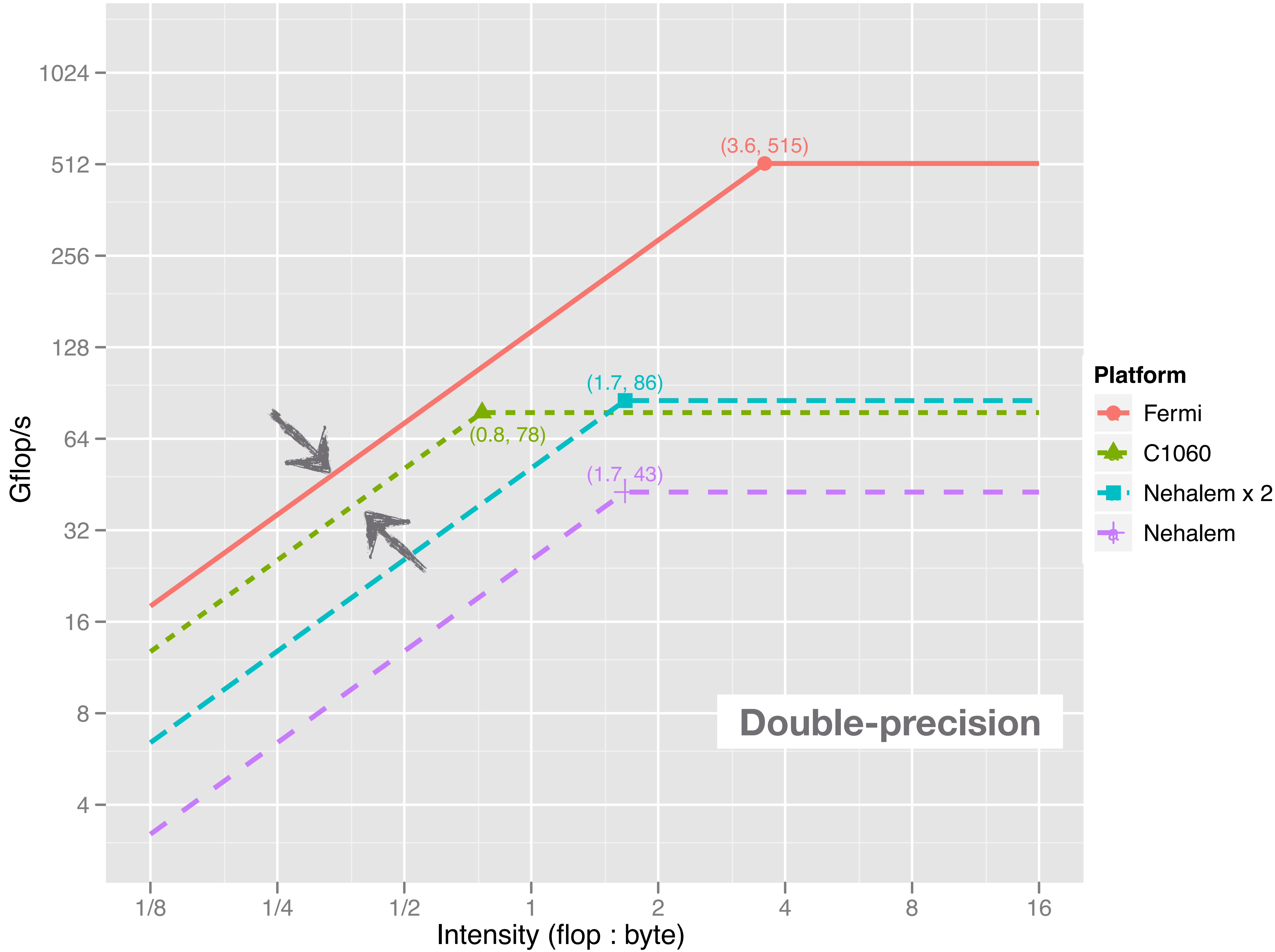

(7.2, 1030)
(9.1, 933)
(3.3, 171)
(3.3, 86)
1024
512
256
128
64
32
16
8
4
Gflop/s
1/8
1/4
1/2
1
2
4
8
16
Intensity (flop : byte)
Platform
Fermi
C1060
Nehalem x 2
Nehalem
Single-precision



(7.2, 1030)
(9.1, 933)
(3.3, 171)
(3.3, 86)
~ 3x
Sparse
mat-vec
~ 4x
Sparse
Direct
~ 6x
FMM
(w/ **Biros**, Vetter;
Gordon Bell finalist)
Single-precision
Platform
Fermi
C1060
Nehalem x 2
Nehalem
Gflop/s
1024
512
256
128
64
32
16
8
4
1/8
1/4
1/2
1
2
4
8
16
Intensity (flop : byte)



(3.6, 515)
~ 6x
(1.7, 86)
(0.8, 78)
(1.7, 43)
~ 3x
Double-precision
Platform
Fermi
C1060
Nehalem x 2
Nehalem
Gflop/s
1024
512
256
128
64
32
16
8
4
1/8
1/4
1/2
1
2
4
8
16
Intensity (flop : byte)



(3.6, 515)
(1.7, 86)
(0.8, 78)
(1.7, 43)
1024
512
256
128
64
32
16
8
4
Gflop/s
1/8
1/4
1/2
1
2
4
8
16
Intensity (flop : byte)
Platform
Fermi
C1060
Nehalem x 2
Nehalem
Double-precision

*Reason 2:* **Productivity: Though there is potential, there is also no free lunch.**

Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage

# Parallel Sorting (survey)

(Does **not** include Merrill & Grimshaw '10)

# Parallel Sorting (survey)

(Does **not** include Merrill & Grimshaw '10)

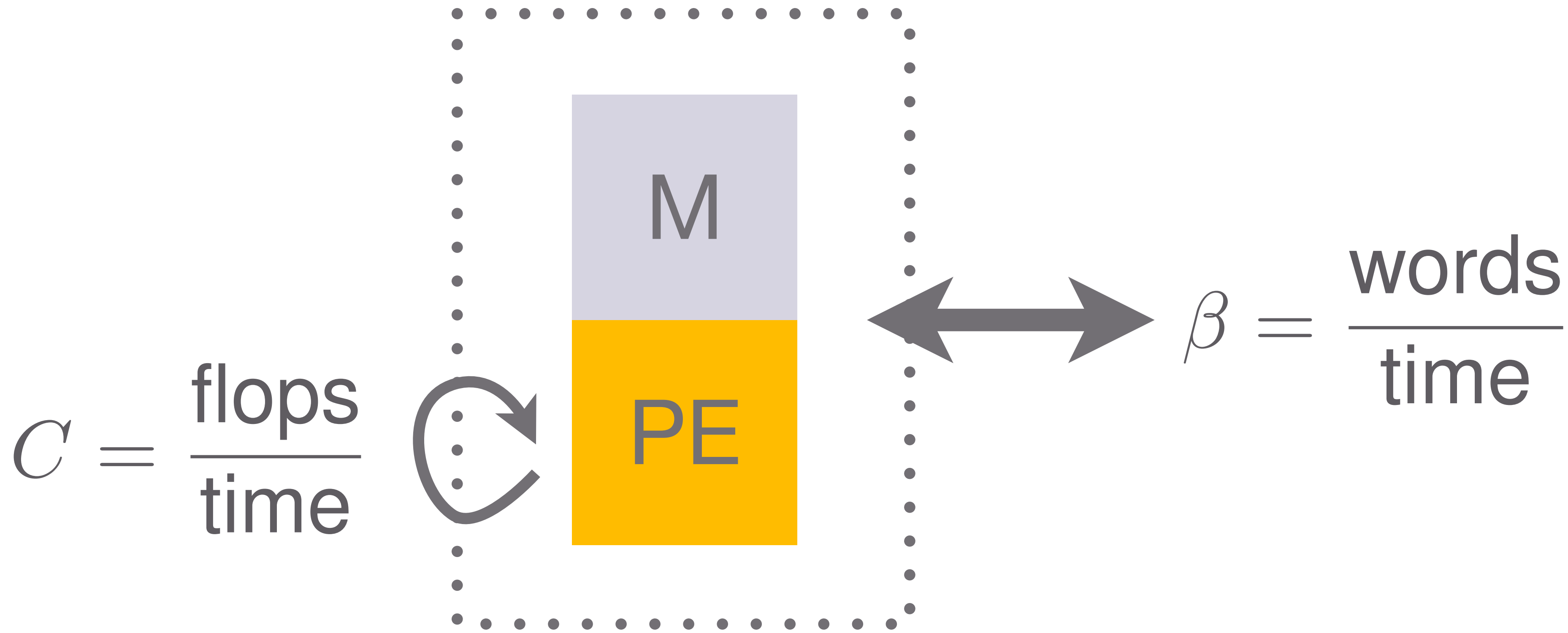
# Parallel Sorting (survey)

(Does **not** include Merrill & Grimshaw '10)

*Reason 3:* **It's a moving target that might end up converging to what we had before.**

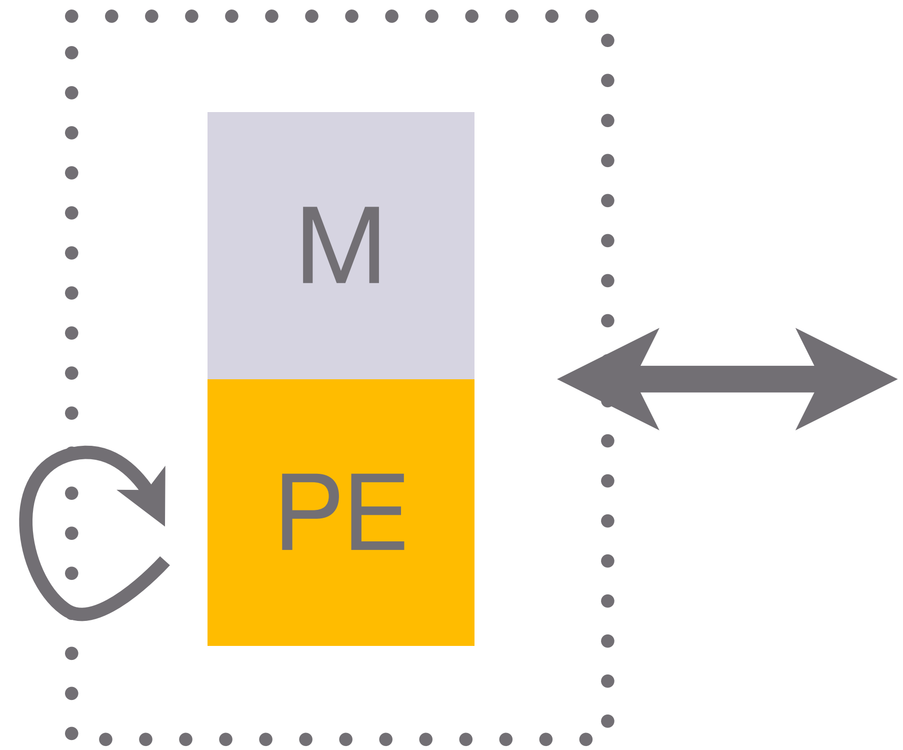
Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage


M
PE
$C = \frac{\text{flops}}{\text{time}}$
$\beta = \frac{\text{words}}{\text{time}}$

HPC 101

$$C = \frac{\text{flops}}{\text{time}}$$




$$\beta = \frac{\text{words}}{\text{time}}$$

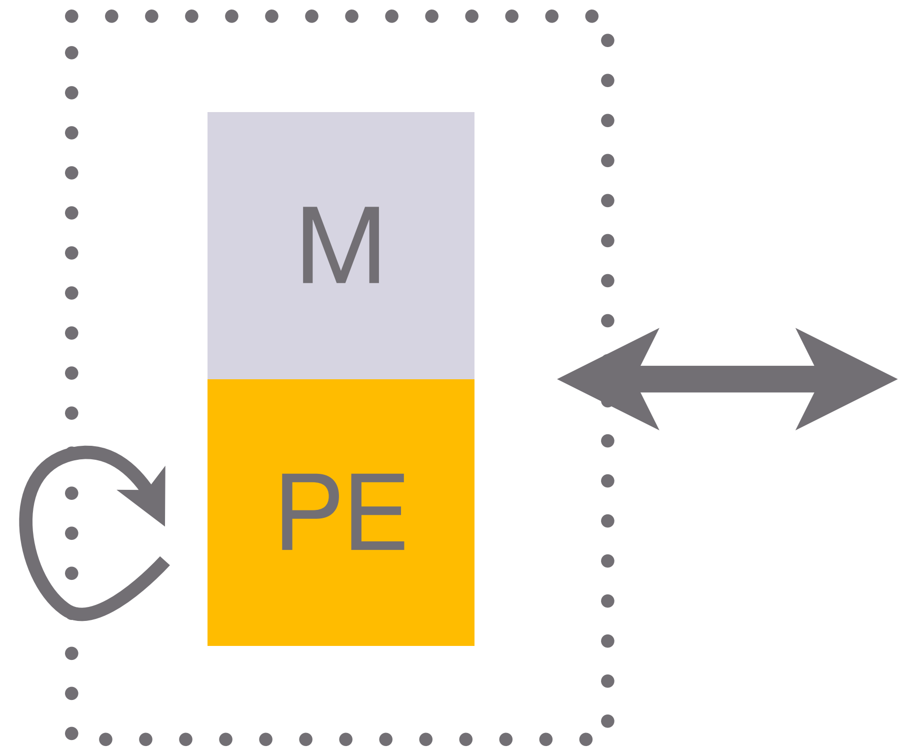
*Balance equation for an I/O-optimal matrix multiply:*

$$\frac{C}{\beta} = \Theta\left(\sqrt{M}\right)$$

## HPC 101

See, for example, Kung (ISCA 1986).

$$C = \frac{\text{flops}}{\text{time}}$$




$$\beta = \frac{\text{words}}{\text{time}}$$

*Balance equation for an I/O-optimal matrix multiply:*

$$\frac{C}{\beta} = \Theta\left(\sqrt{M}\right)$$

*For comparison-based sort:*

$$\frac{C}{\beta} = \Theta\left(\log_2 M\right)$$

# HPC 101

See, for example, Kung (ISCA 1986).

Having said all that, I am still **optimistic** about the role GPUs will play in current and future systems!

Georgia Tech College of Computing
Computational Science and Engineering

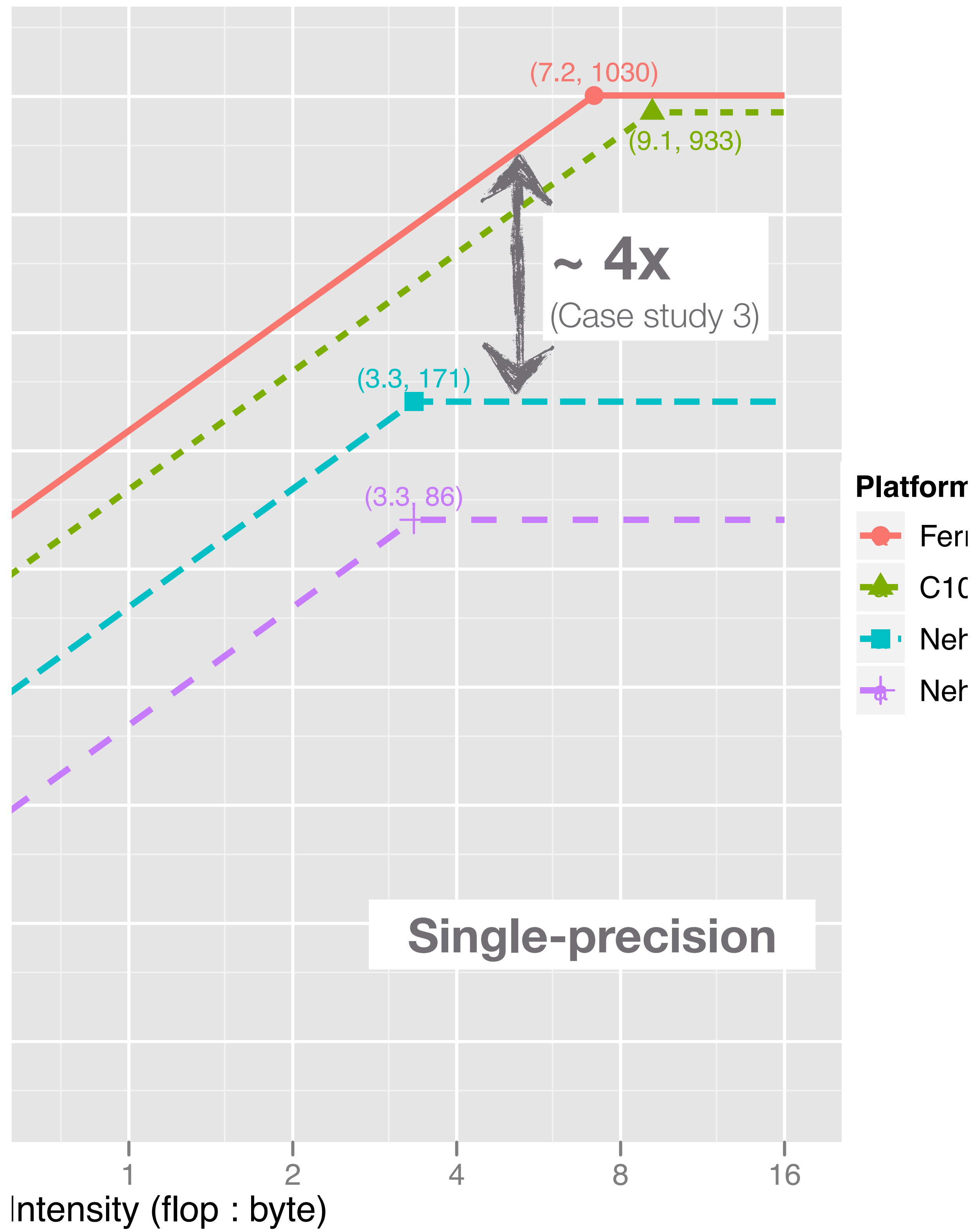
hpcgarage

# Sparse direct solvers

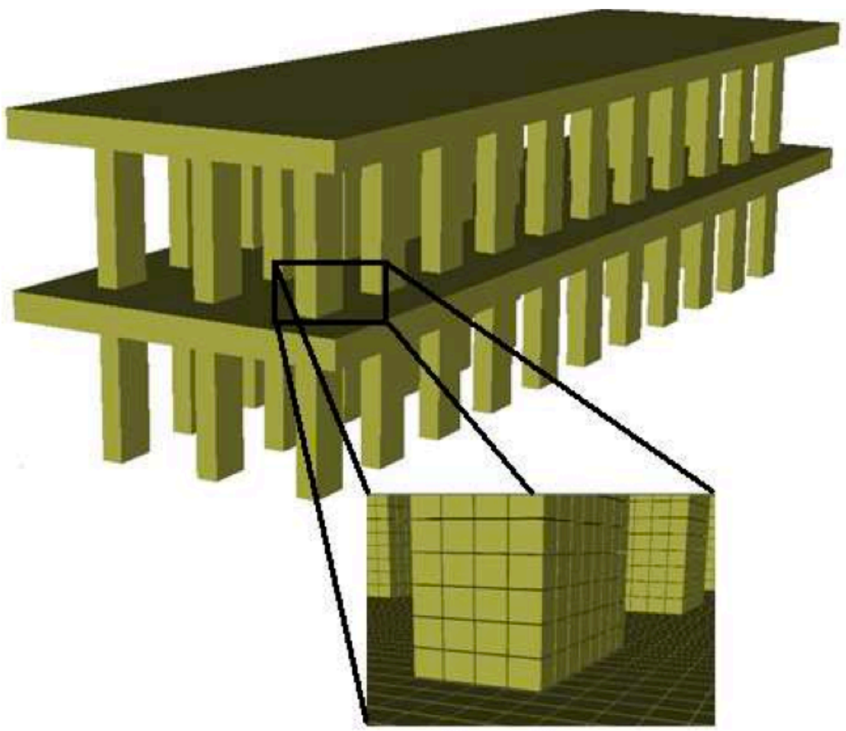
## Kent Czechowski, M. Efe Guney, R. Vuduc

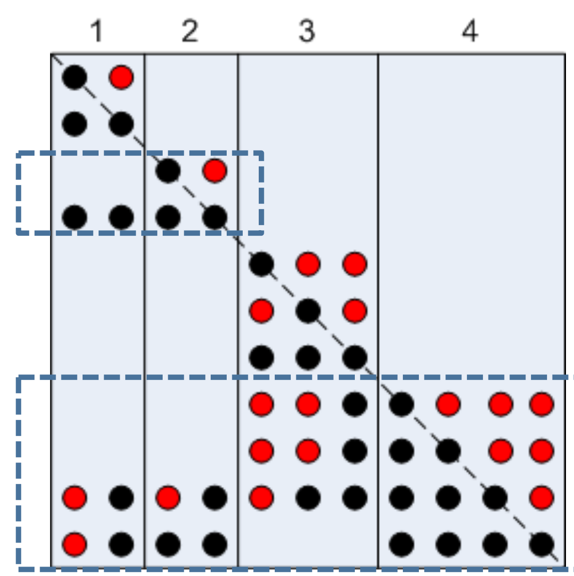
**(Work in progress)**

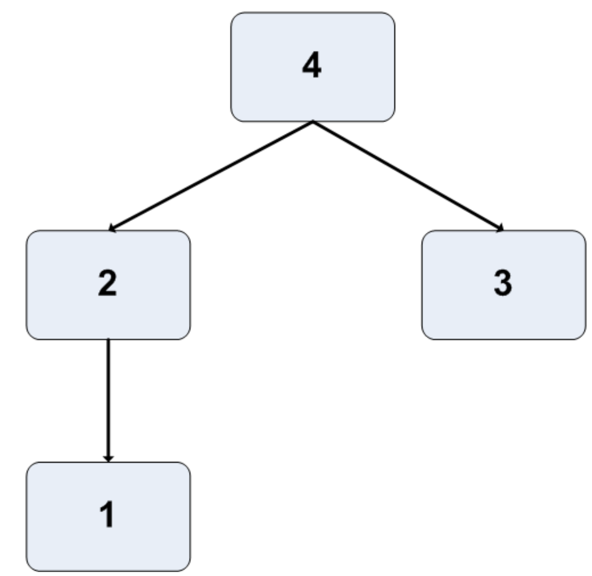
M. Efe Guney. *High-performance direct solution of finite-element problems on multi-core processors*. Ph.D. Thesis, School of Civil Engineering, Georgia Tech, May 2010.



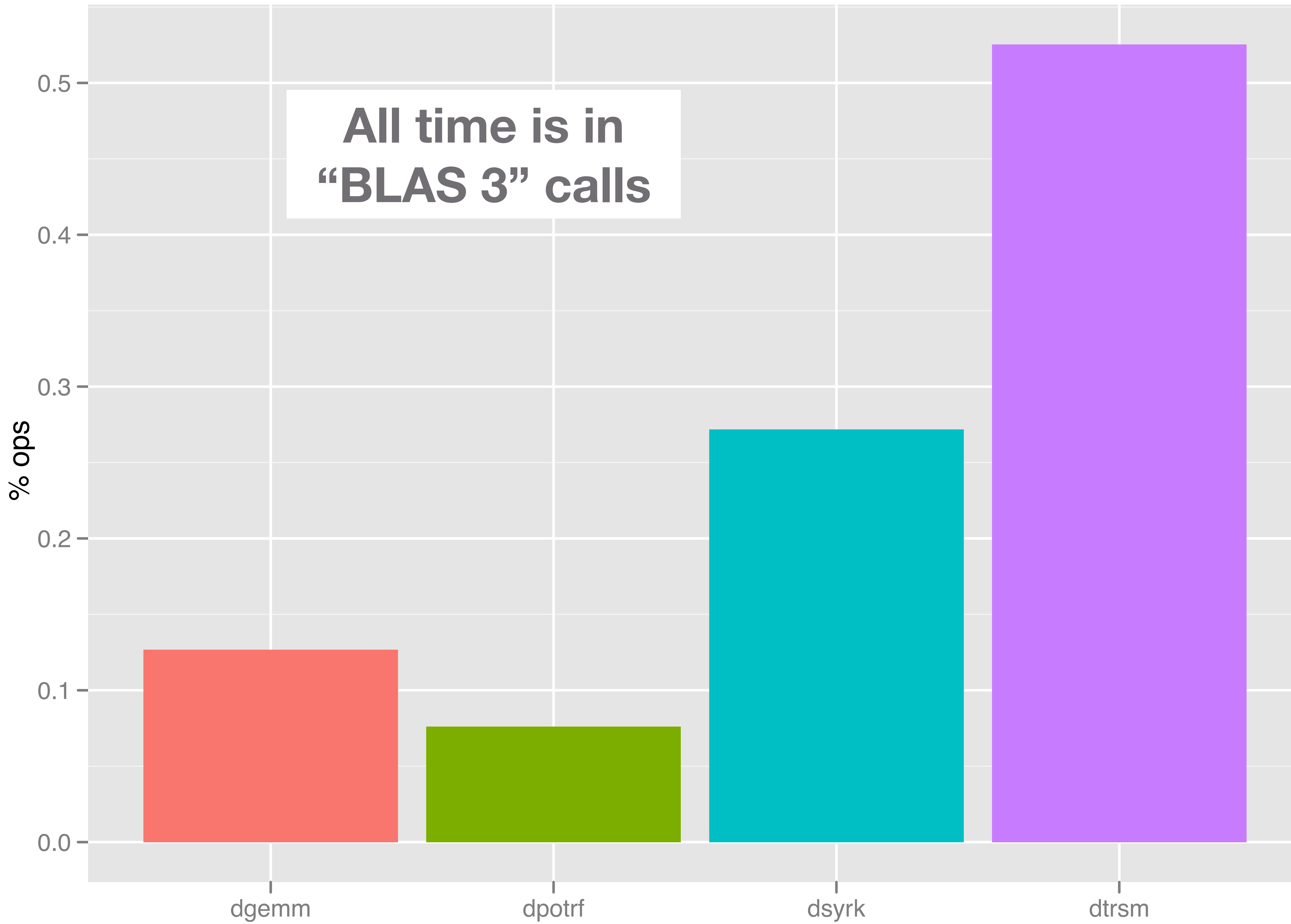
# Anatomy of a sparse direct solver

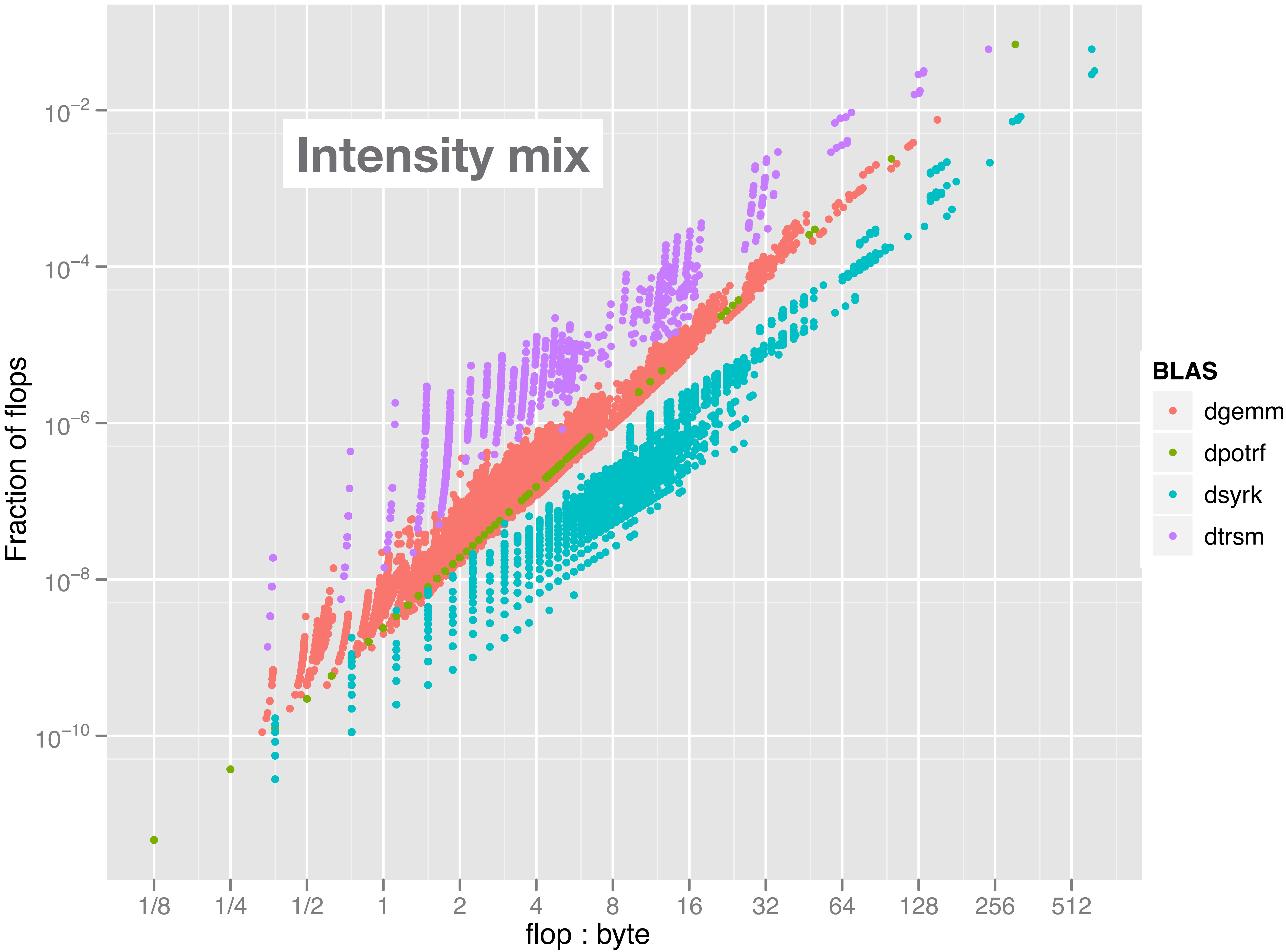


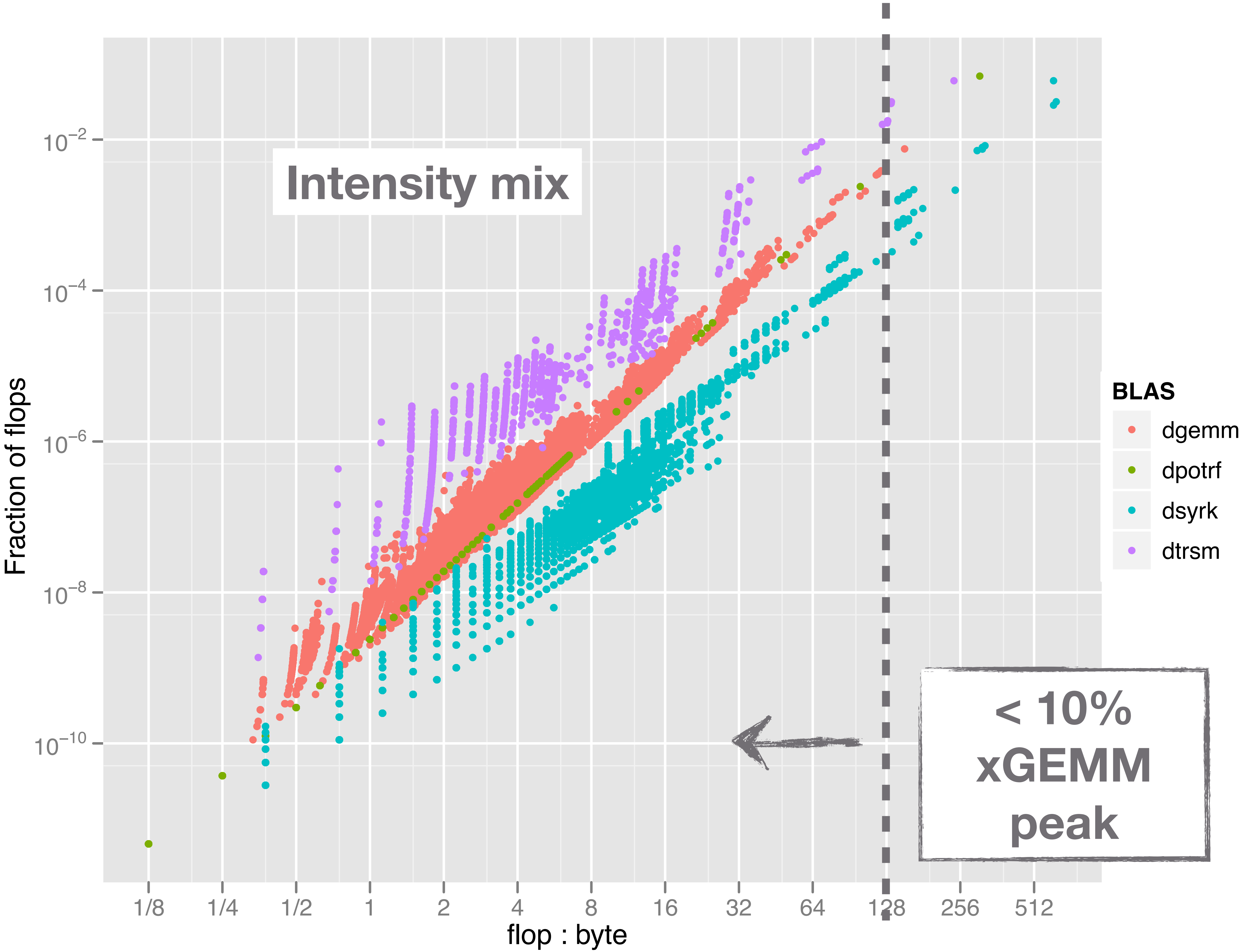


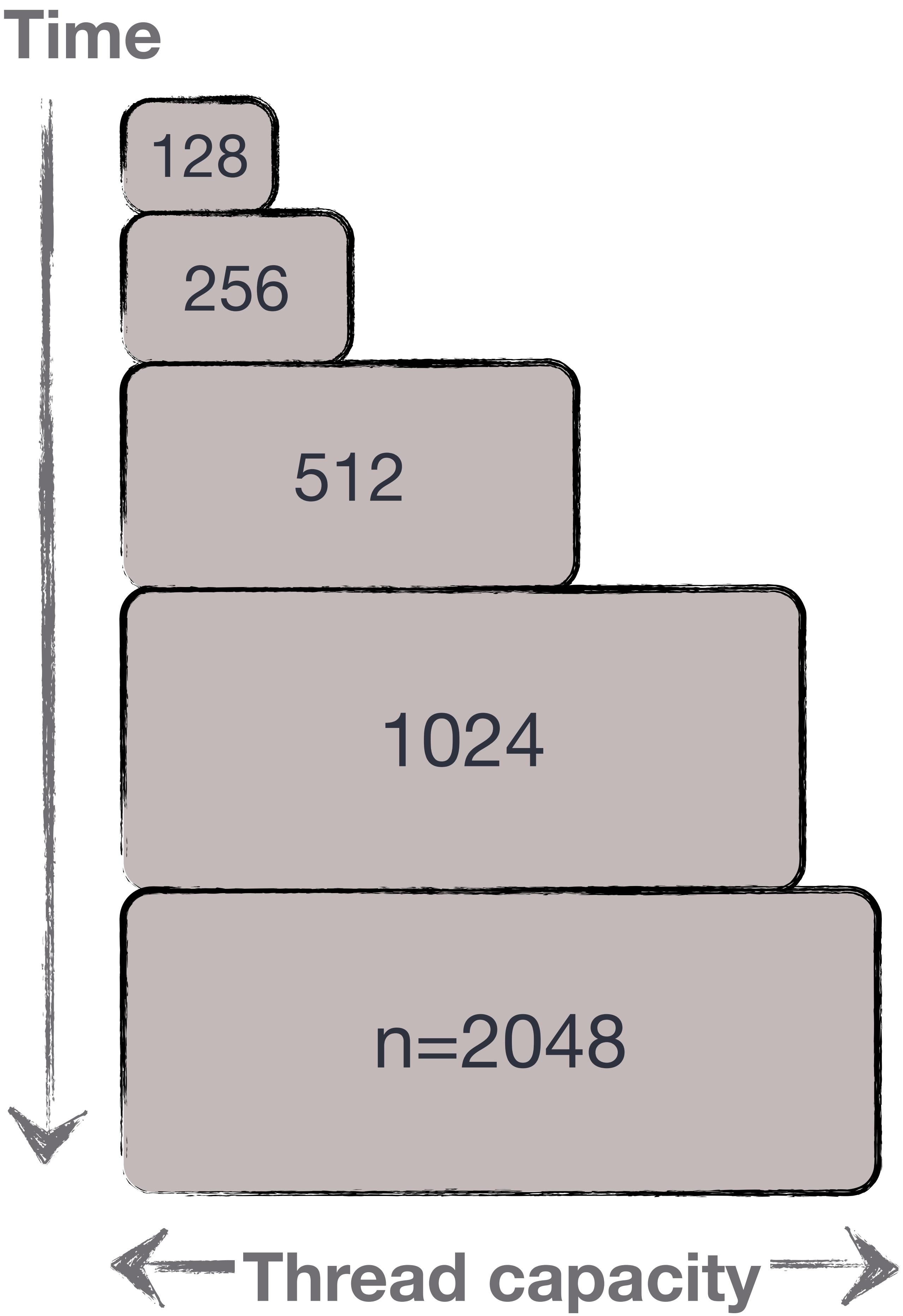



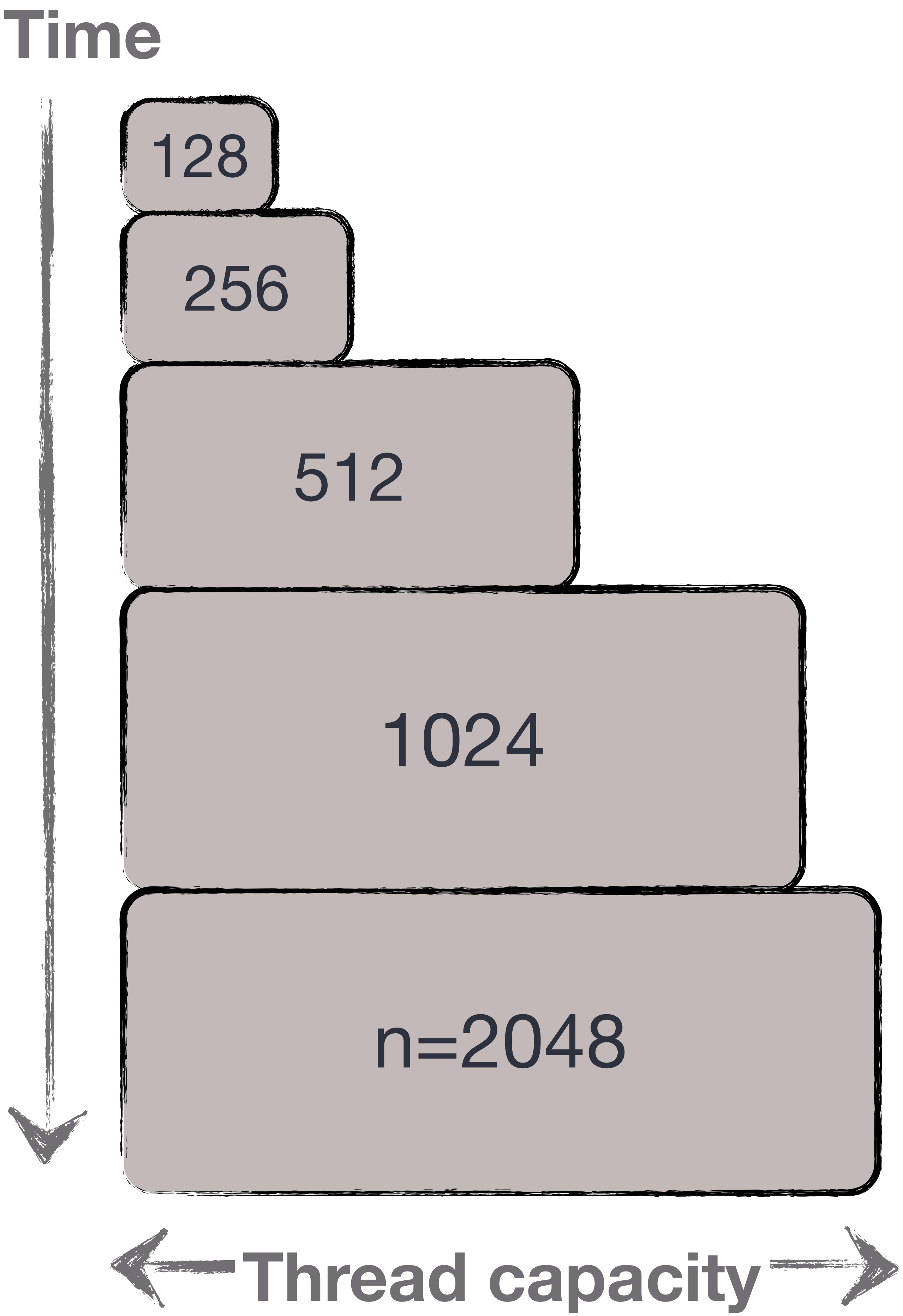

- Sparse Cholesky factorization, $A = L \cdot L^T$, where $A$ & $L$ are sparse
- Mixed compute intensity, average of ~ 4 flops : byte for sample problem

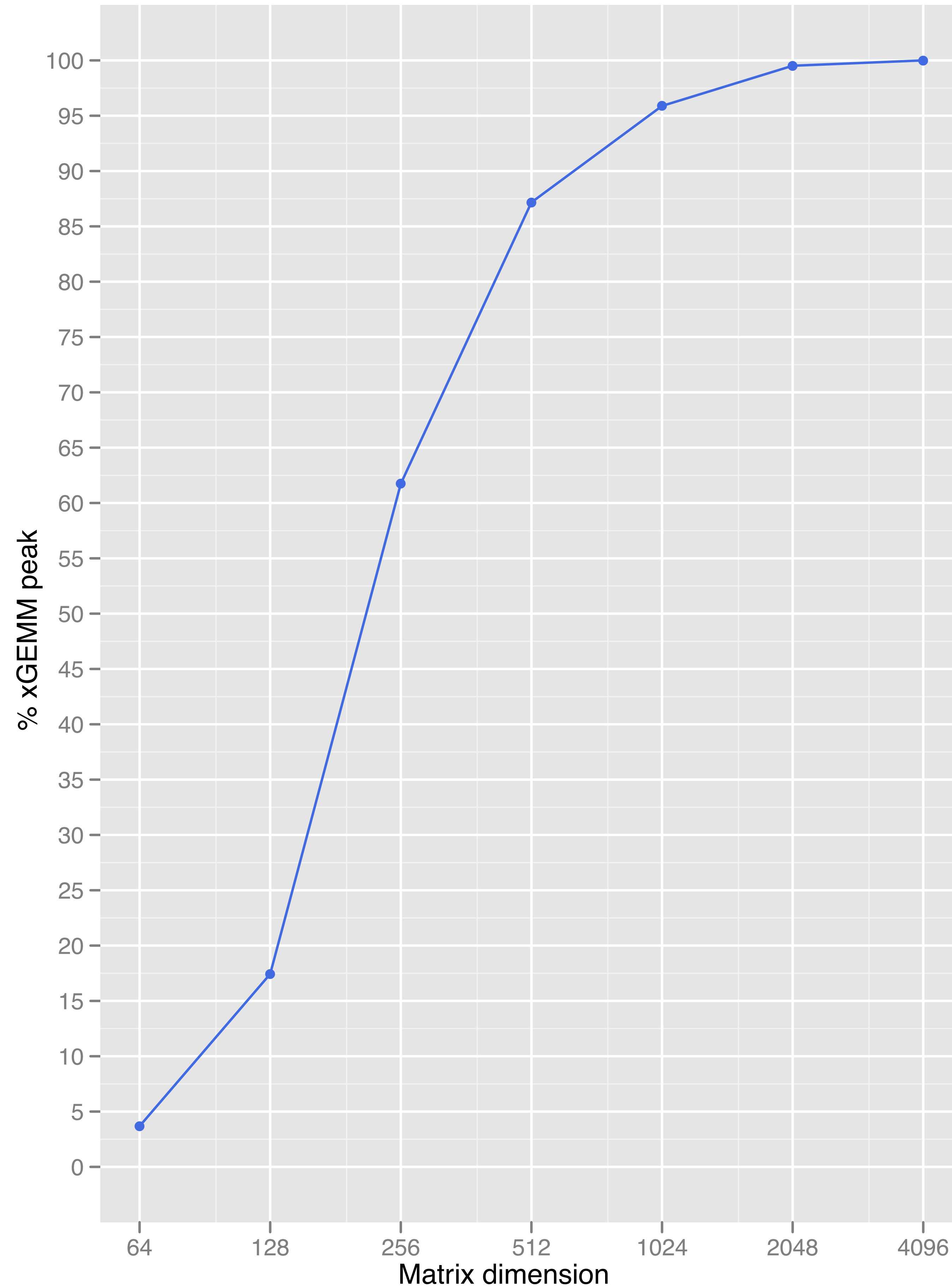

All time is in
“BLAS 3” calls
% ops
0.0
0.1
0.2
0.3
0.4
0.5
dgemm
dpotrf
dsyrk
dtrsm

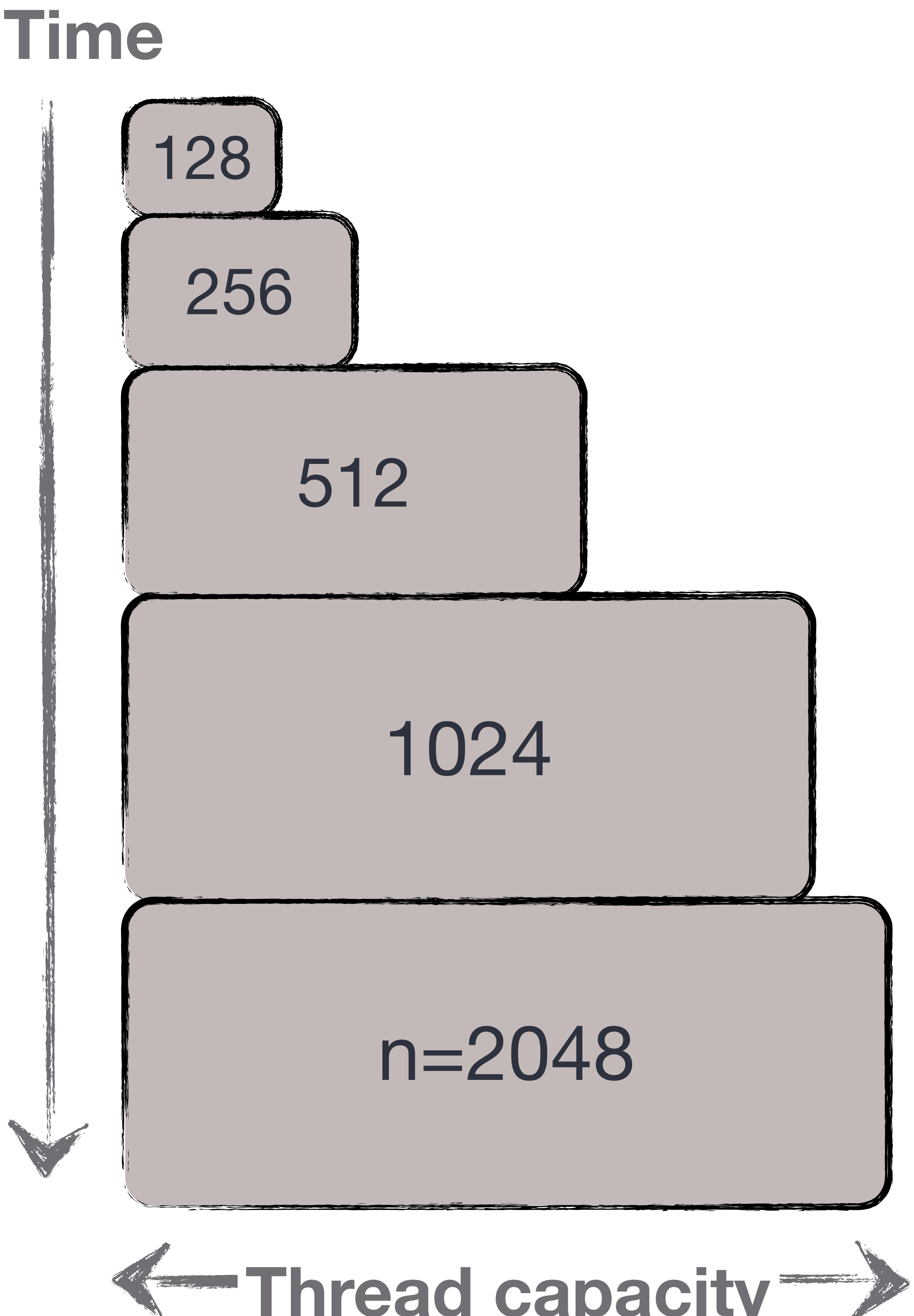

Intensity mix
Fraction of flops
$10^{-2}$
$10^{-4}$
$10^{-6}$
$10^{-8}$
$10^{-10}$
flop : byte
1/8
1/4
1/2
1
2
4
8
16
32
64
128
256
512
BLAS
dgemm
dpotrf
dsyrk
dtrsm

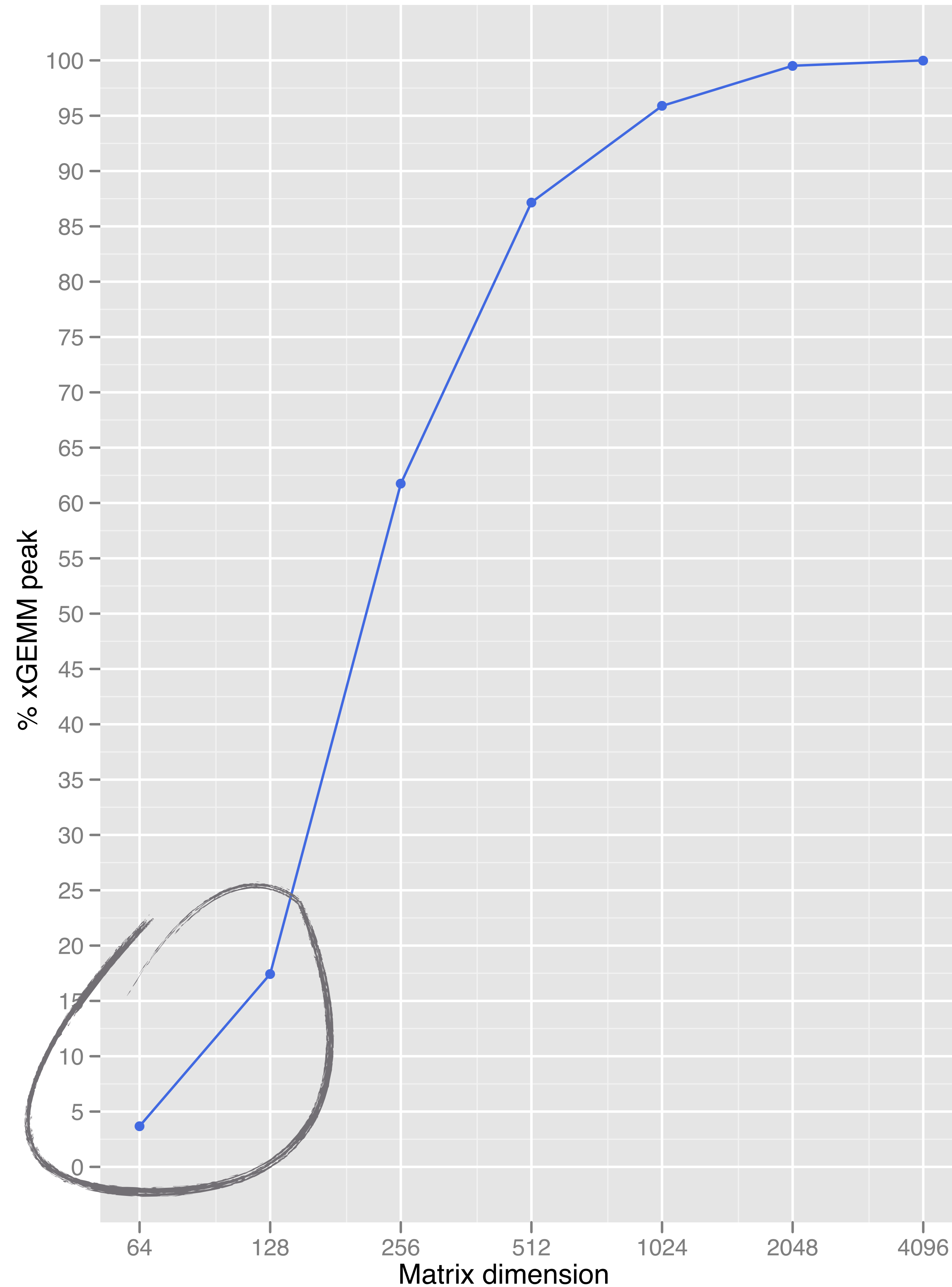

Intensity mix
Fraction of flops
10−2
10−4
10−6
10−8
10−10
flop : byte
1/8
1/4
1/2
1
2
4
8
16
32
64
128
256
512
BLAS
dgemm
dpotrf
dsyrk
dtrsm
< 10% xGEMM peak

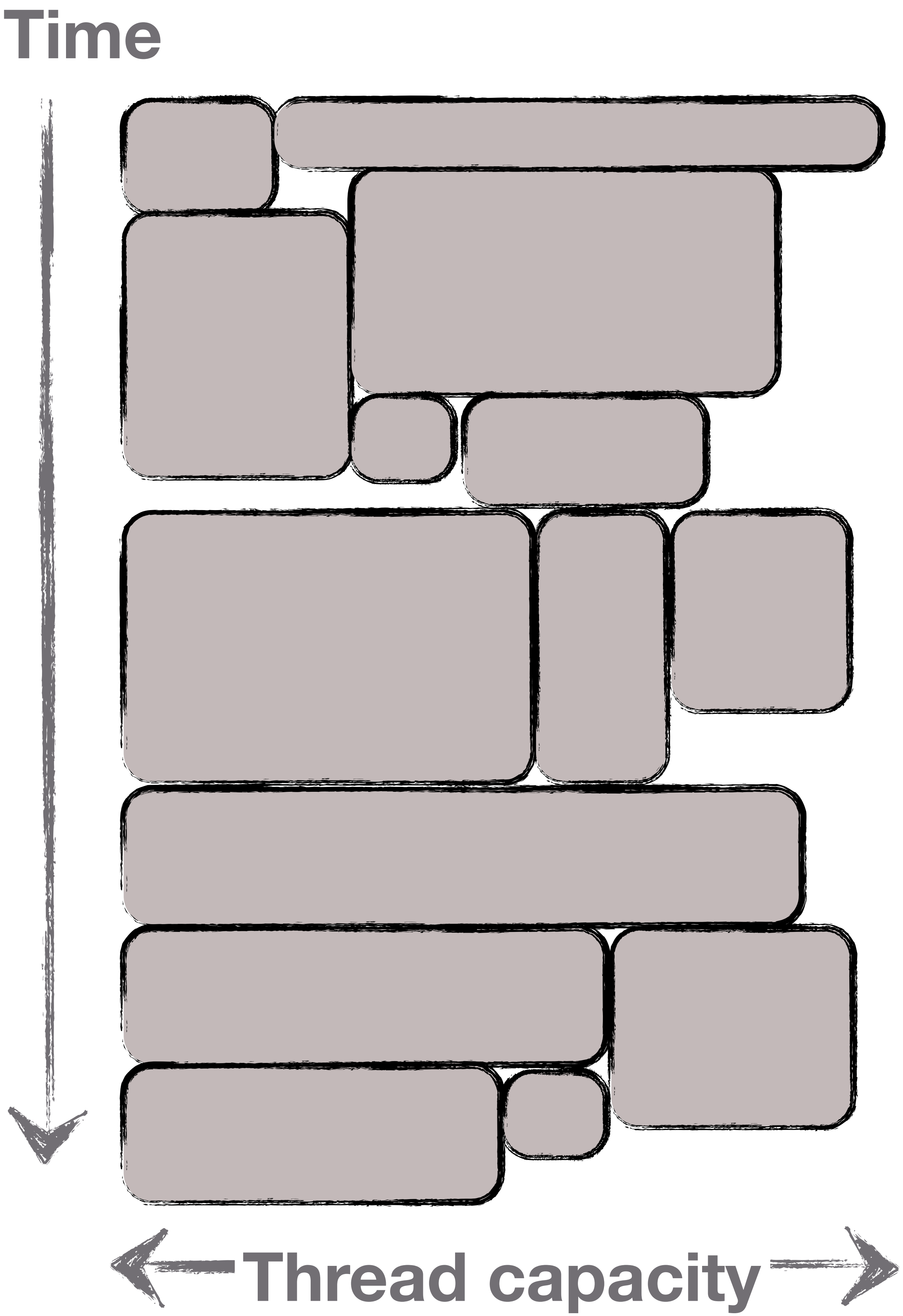

Time
128
256
512
1024
n=2048
Thread capacity


Time
128
256
512
1024
n=2048
Thread capacity

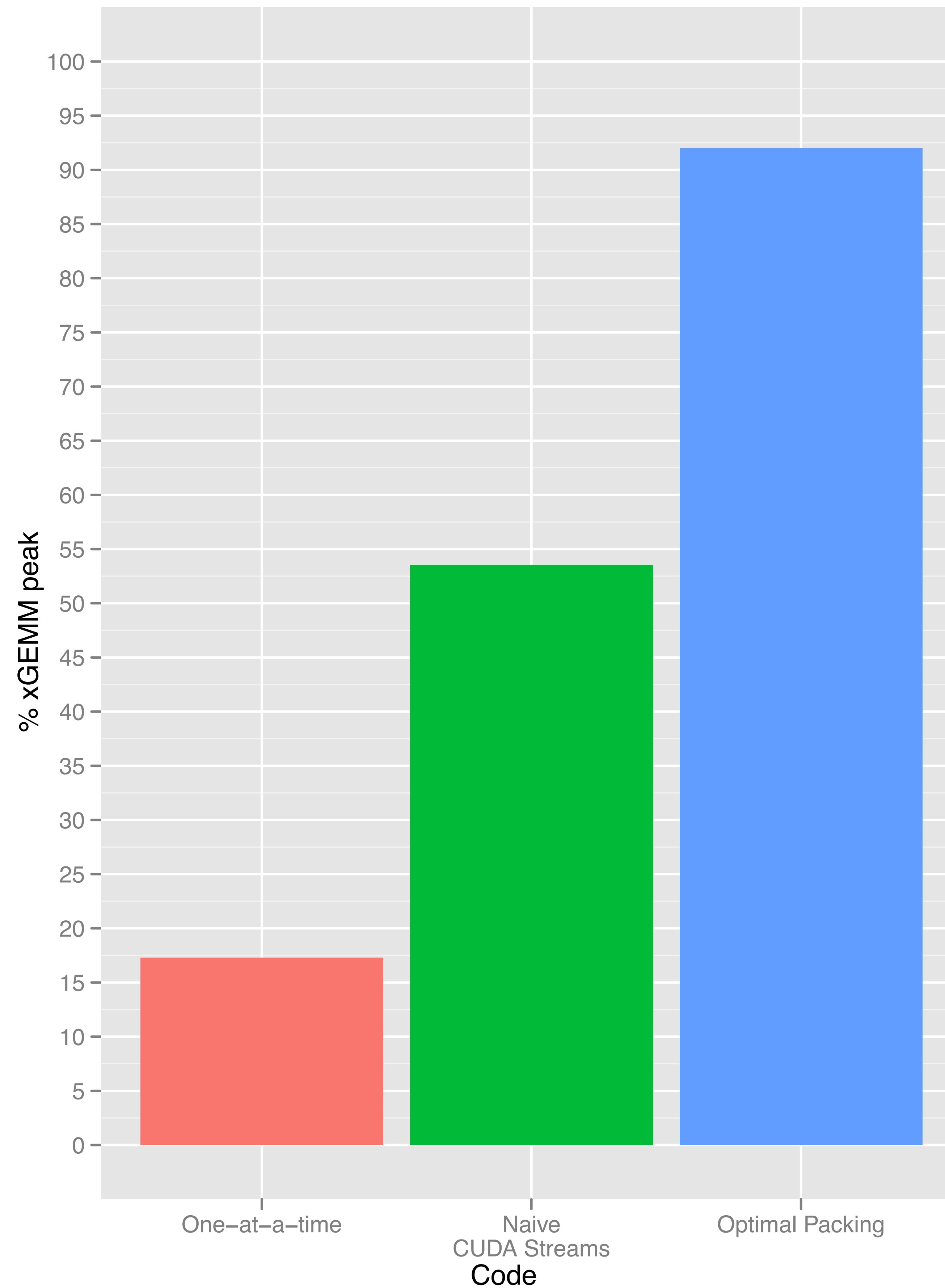

% xGEMM peak
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
64
128
256
512
1024
2048
4096
Matrix dimension


Time
128
256
512
1024
n=2048
Thread capacity

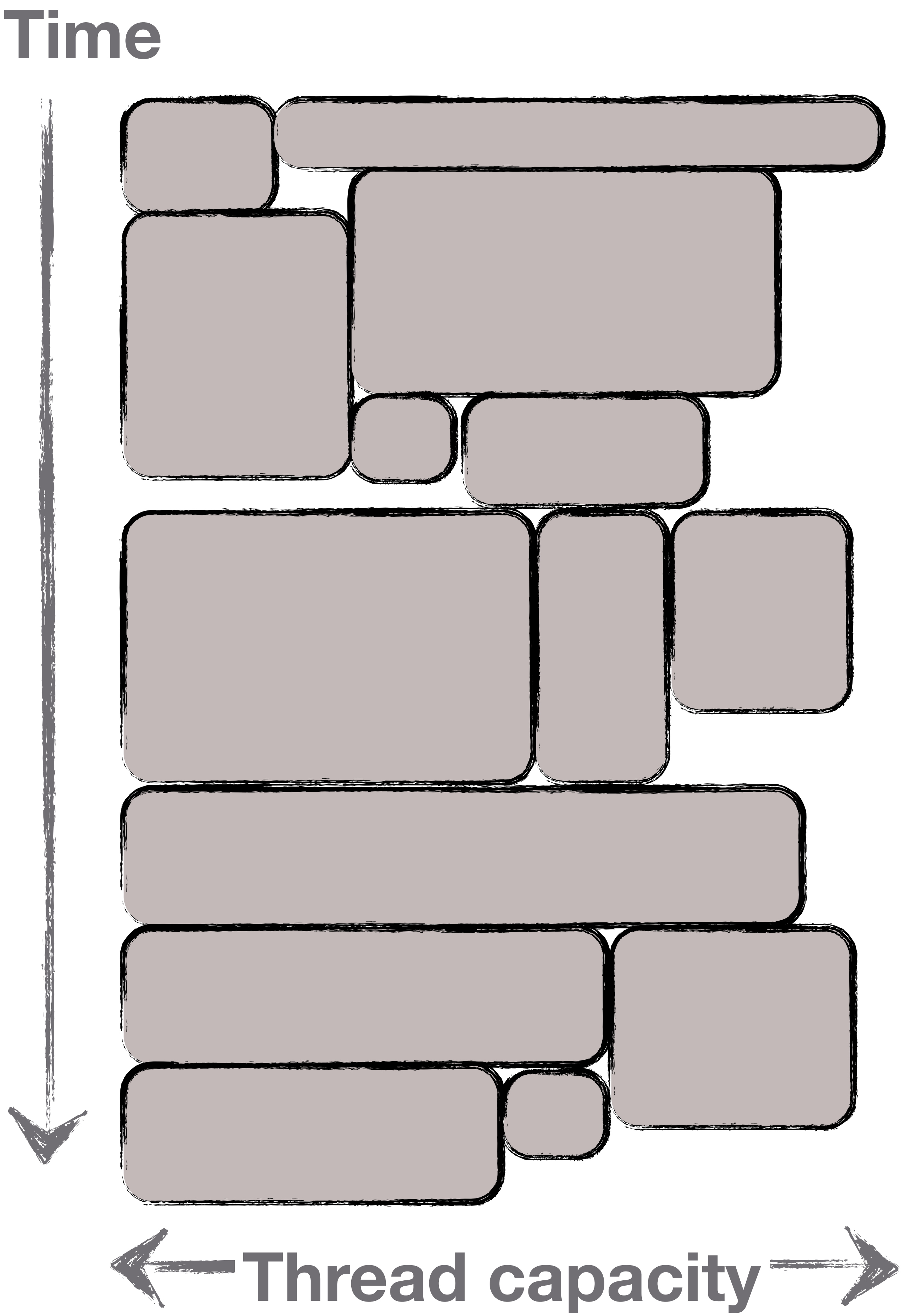

100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
% xGEMM peak
64
128
256
512
1024
2048
4096
Matrix dimension

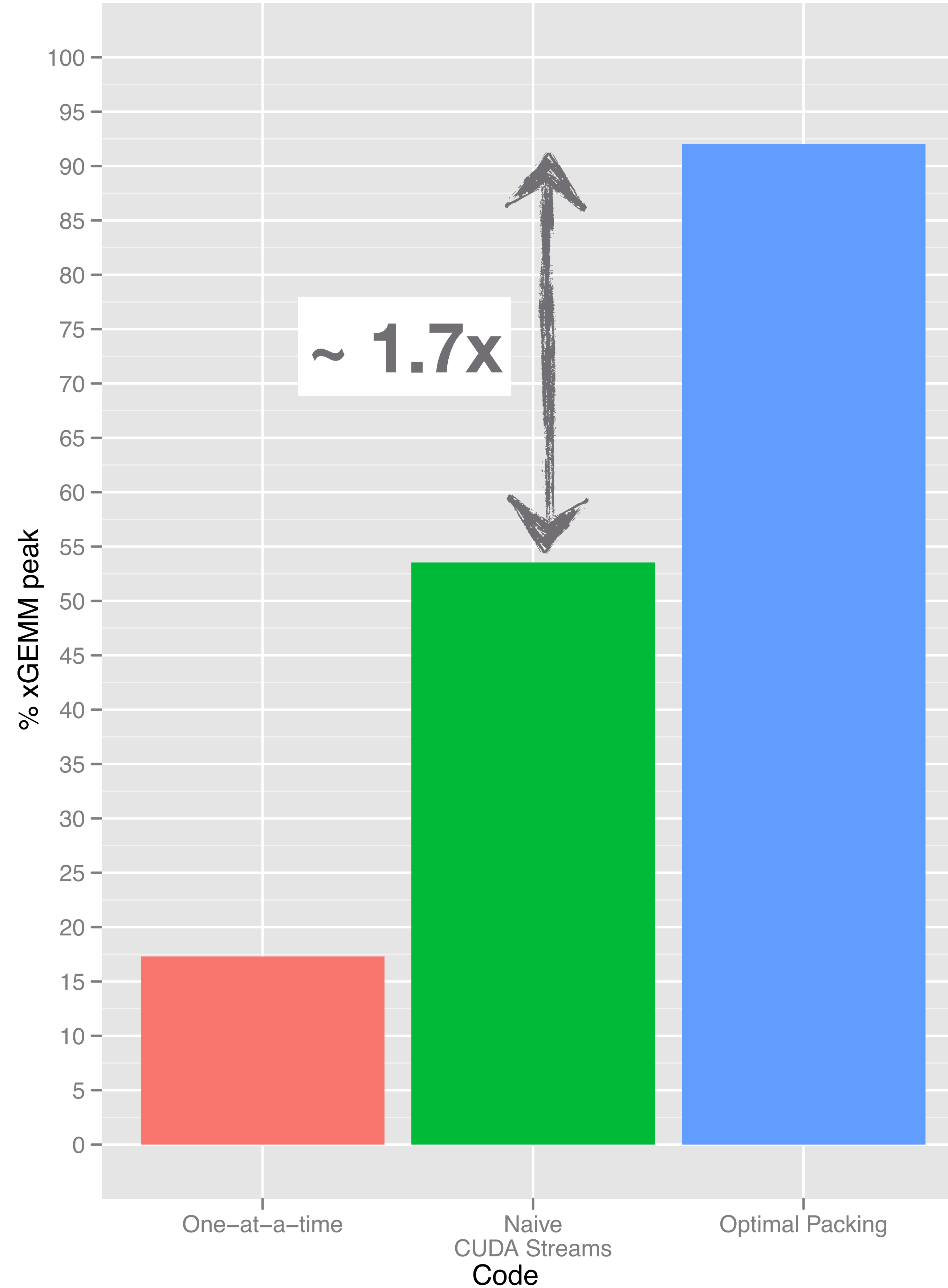


Time
Thread capacity

% xGEMM peak
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
One–at–a–time
Naive
CUDA Streams
Optimal Packing
Code


Time
Thread capacity

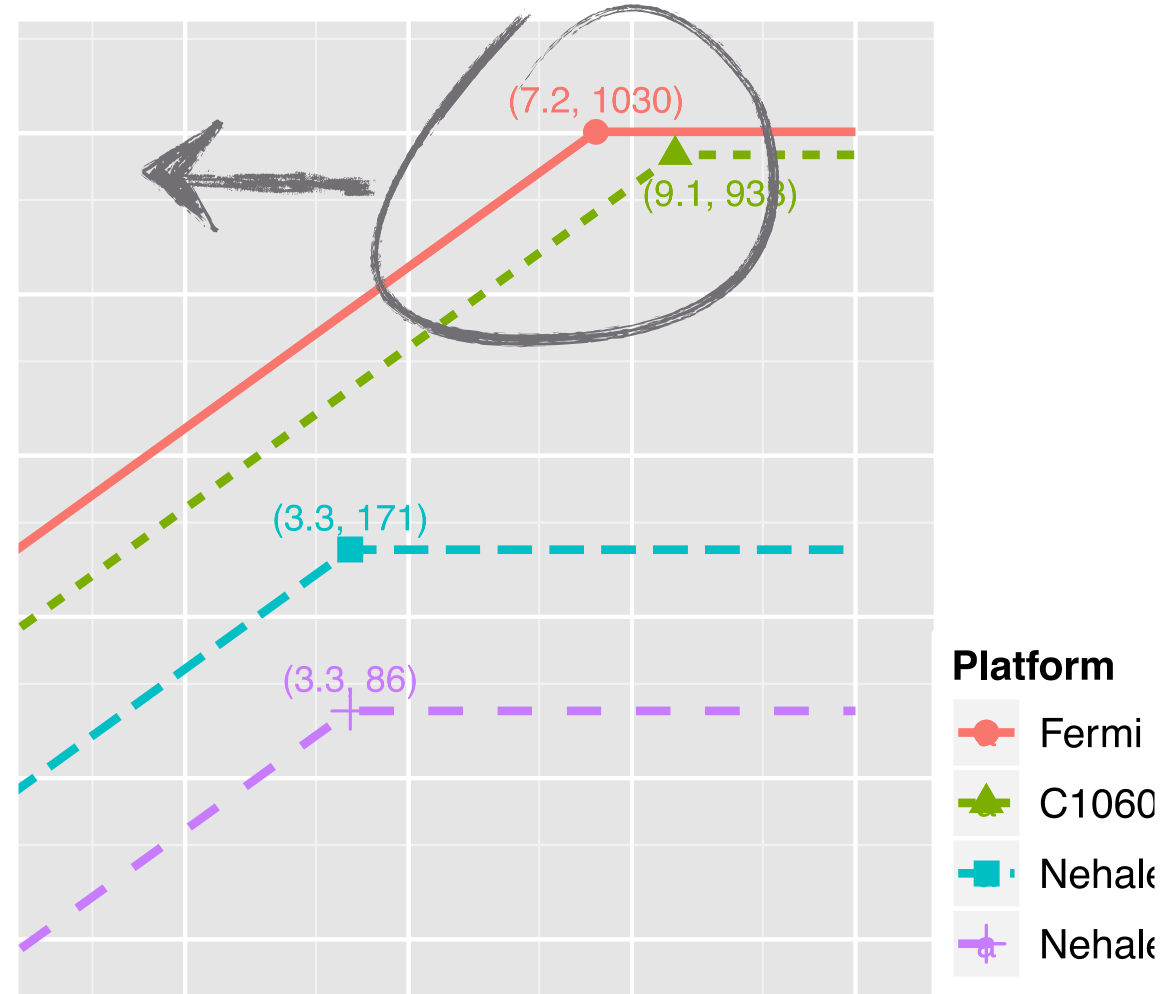

~ 1.7x
% xGEMM peak
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
One–at–a–time
Naive
CUDA Streams
Optimal Packing
Code

*Some concluding questions…*

* What is the right way to think about **opportunity cost**?

* What are the **performance principles** for reducing tuning effort?

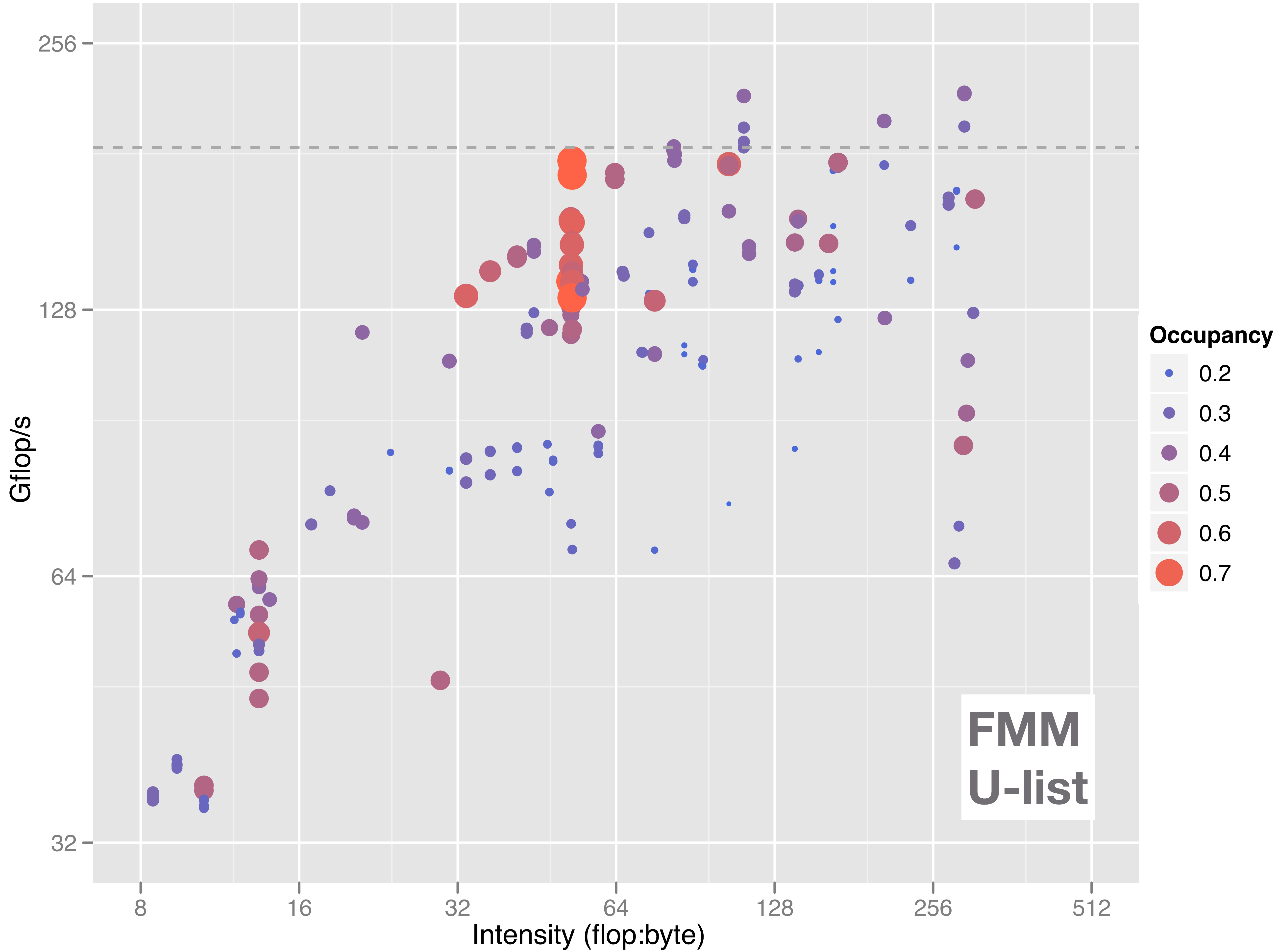
* What **applications** will lead to better designs?



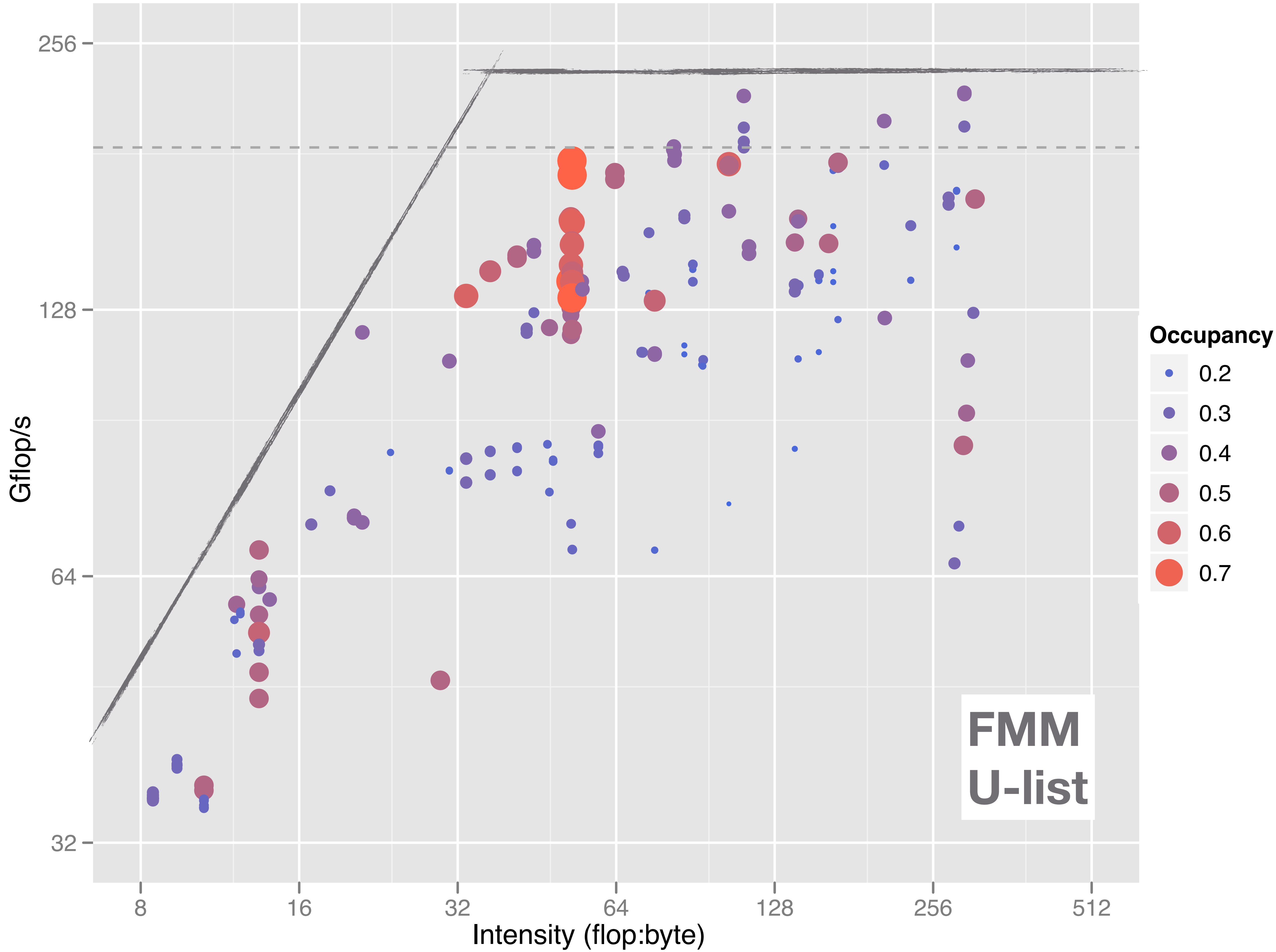

Georgia Tech College of Computing
Computational Science and Engineering

hpcgarage

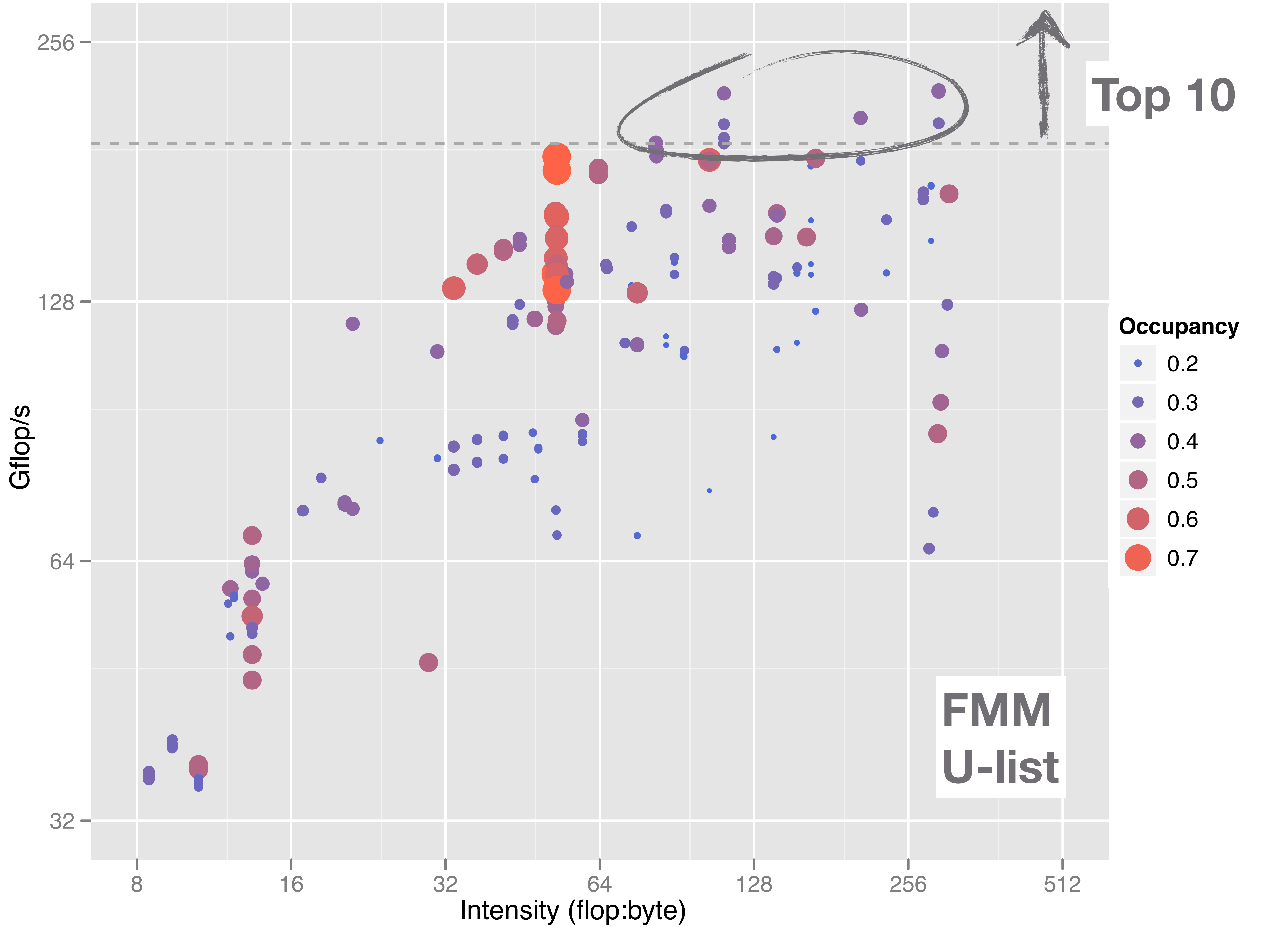
- Backup slides



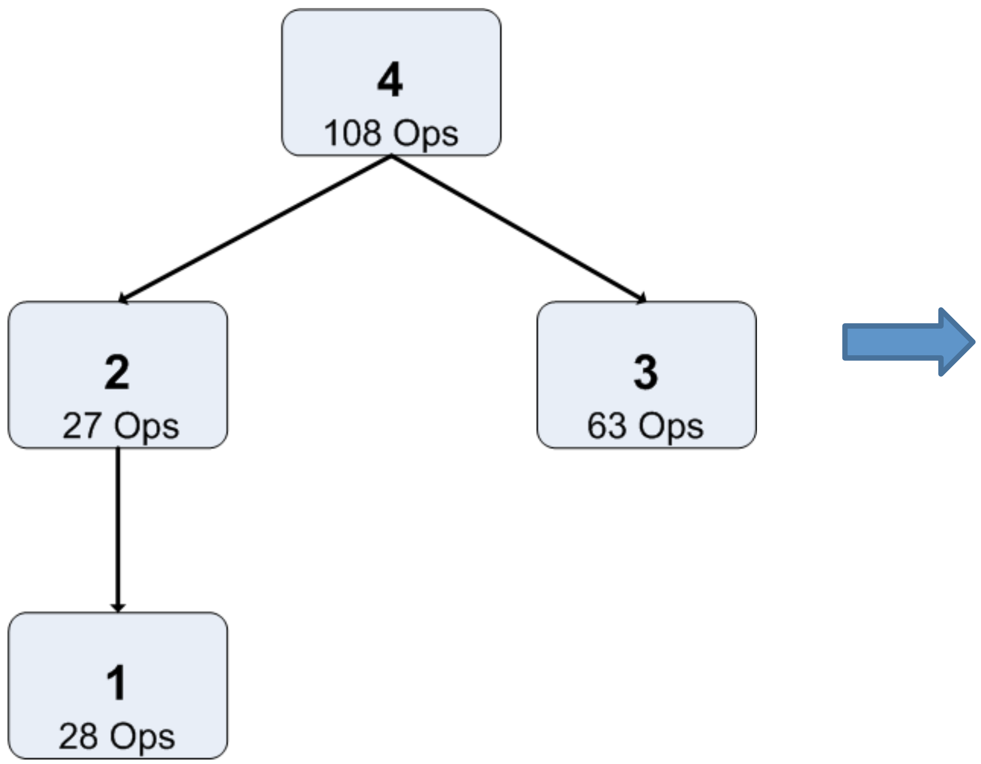
256
128
64
32
Gflop/s
8
16
32
64
128
256
512
Intensity (flop:byte)
Occupancy
0.2
0.3
0.4
0.5
0.6
0.7
FMM
U-list

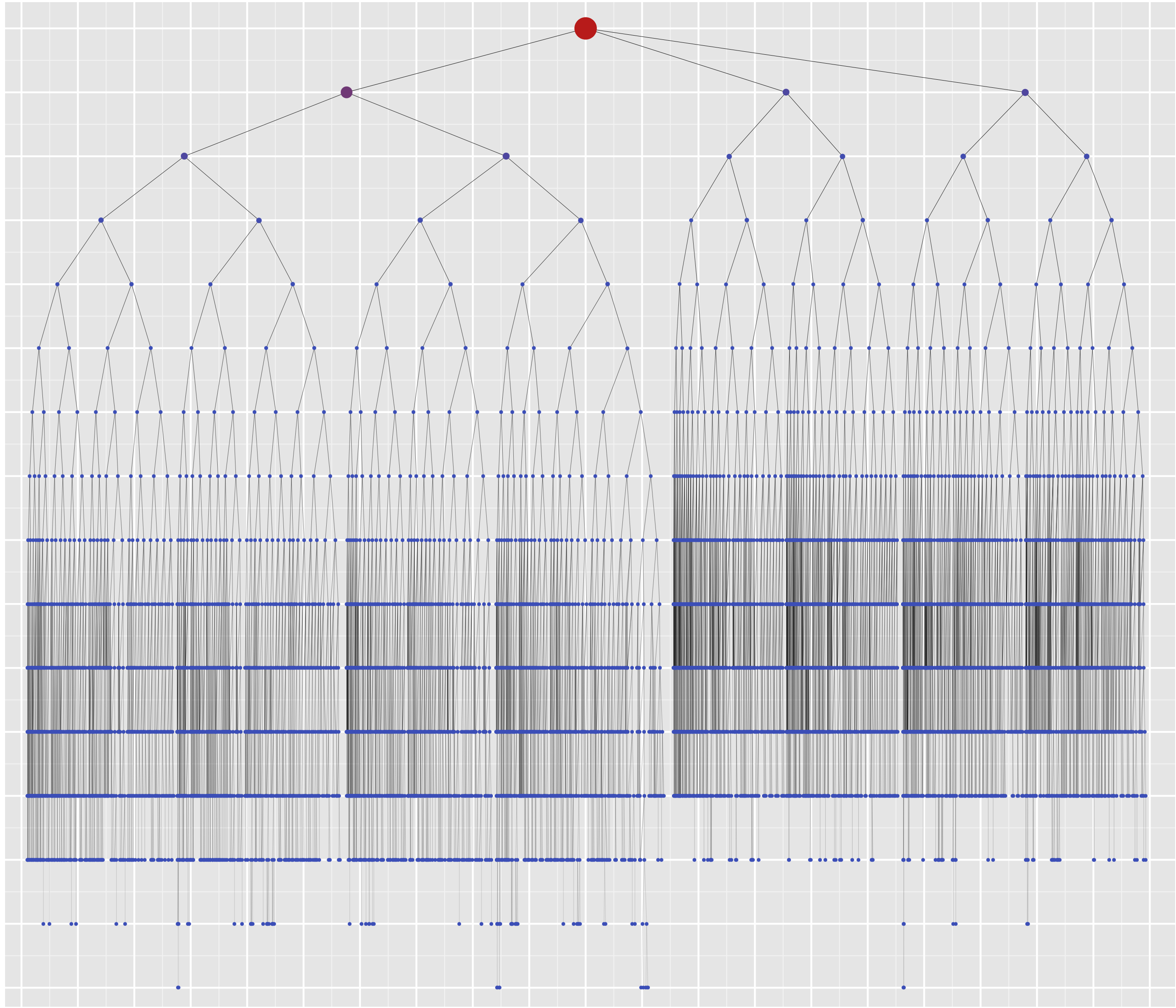


256
128
64
32
Gflop/s
8
16
32
64
128
256
512
Intensity (flop:byte)
Occupancy
0.2
0.3
0.4
0.5
0.6
0.7
FMM
U-list



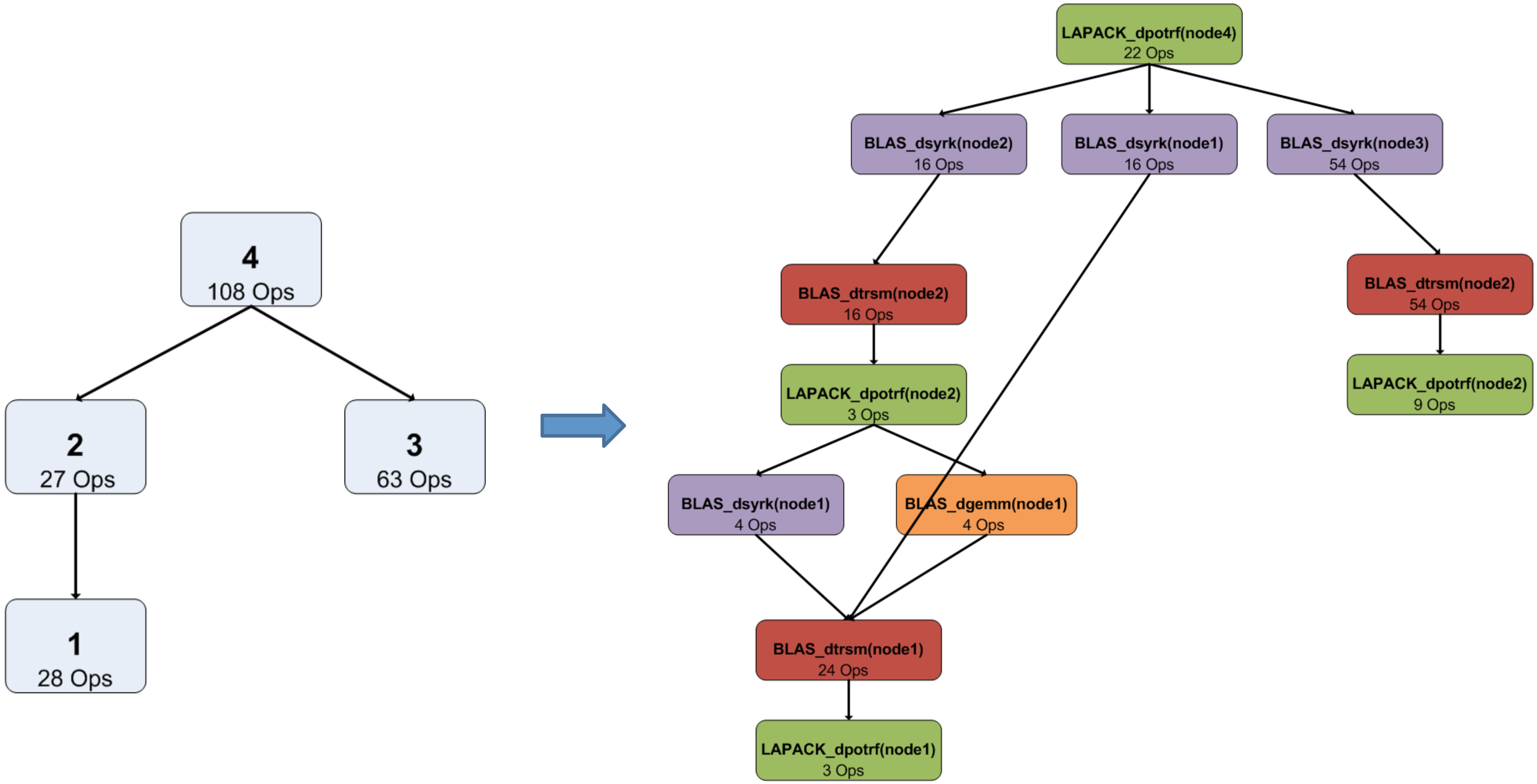
256
128
64
32
8
16
32
64
128
256
512
Gflop/s
Intensity (flop:byte)
Top 10
Occupancy
0.2
0.3
0.4
0.5
0.6
0.7
FMM
U-list



# Parallelism = Elimination tree

Independent subtrees may be processed in parallel.




# Finer-grained dependencies

Colored circles on the right are BLAS calls on operands of varying size.