Georgia Tech College of Computing
Computational Science and Engineering

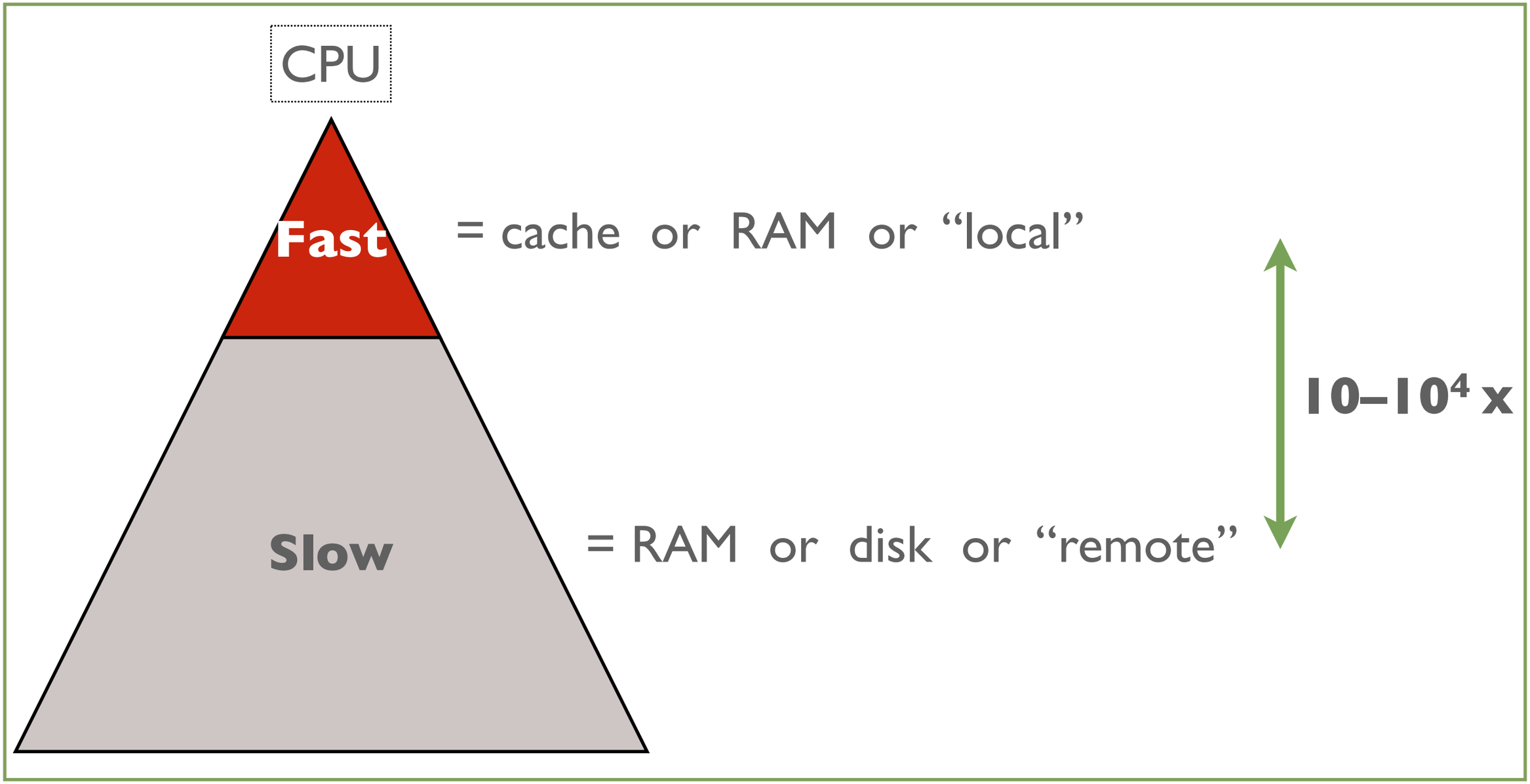
Memory hierarchies (1/2)

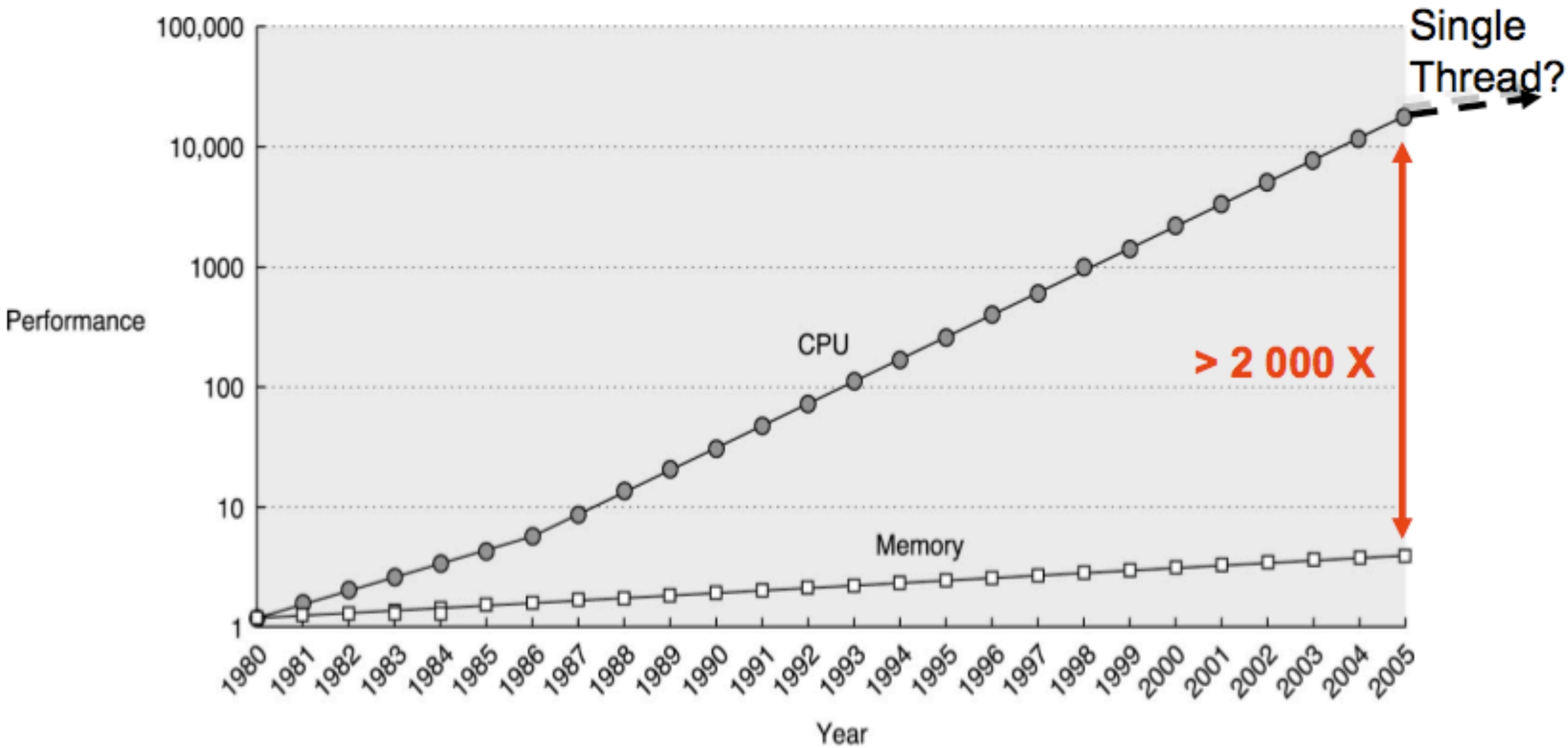
Prof. Rich Vuduc <richie@cc>

Feb. 18, 2009

- Single-processor (single-core) performance:
  A motivating example

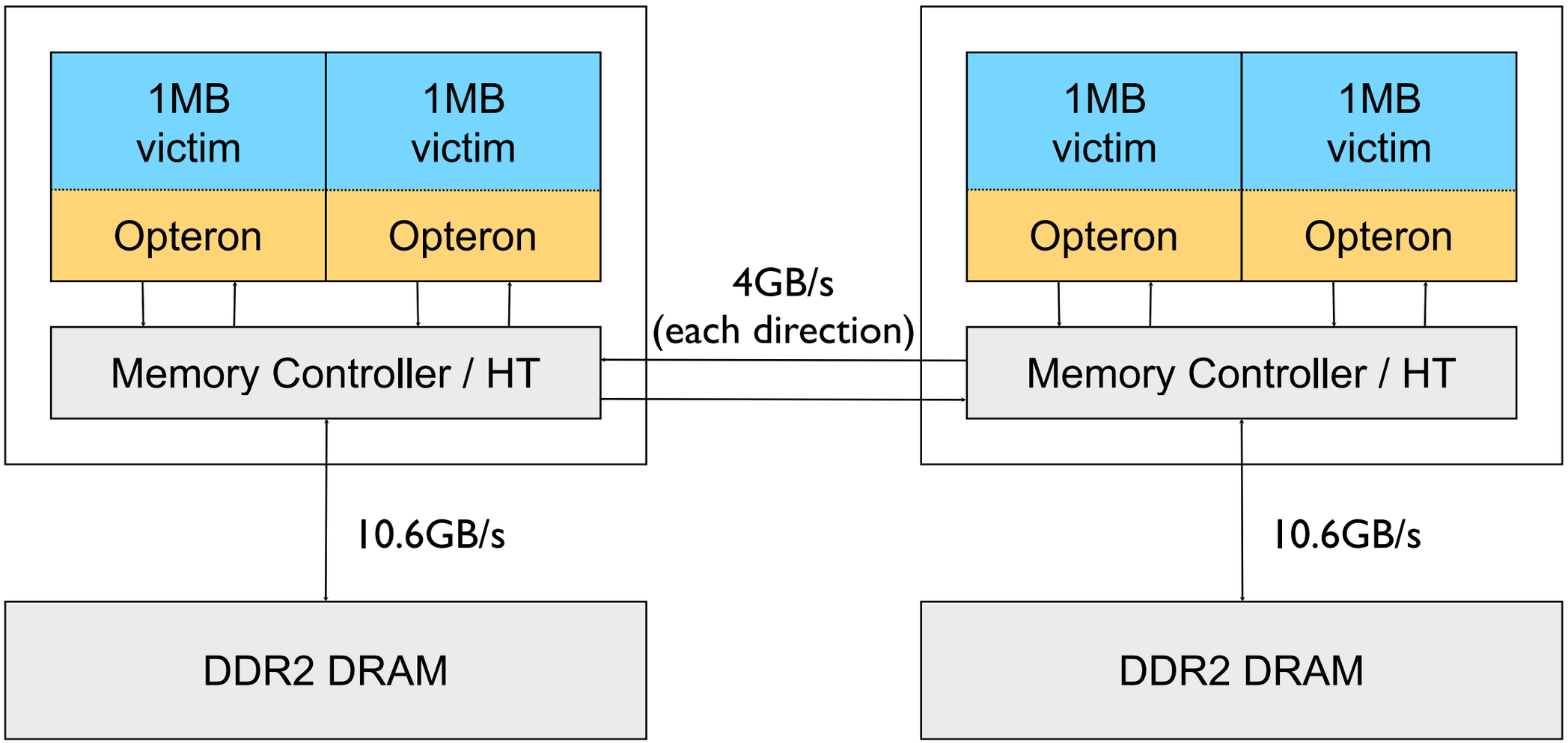
Architecture 101: Memory hierarchy.





Processor-memory gap.

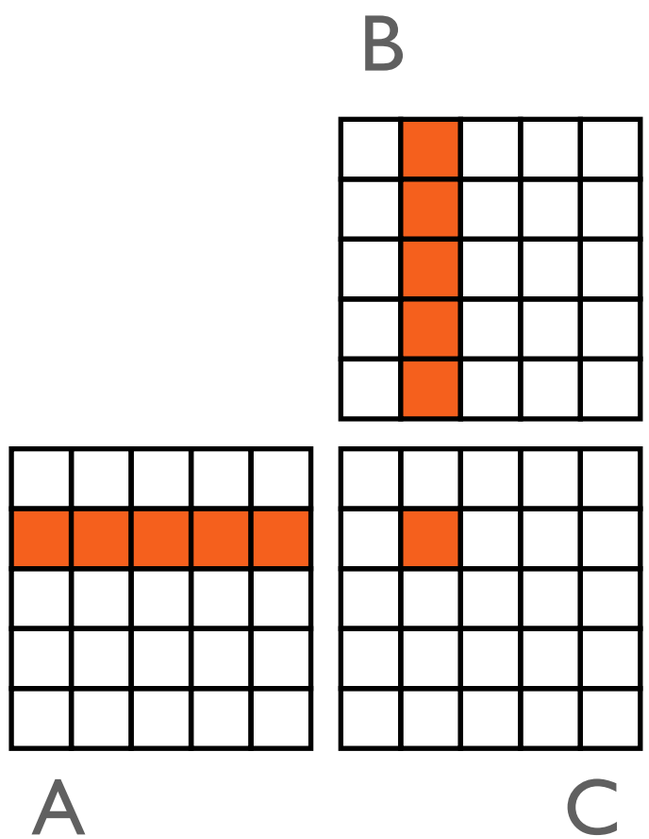
Memory hierarchies reflect and help deal with growing speed gap between processors & memory.



AMD Opteron X2 (2 x 2)

Linear algebra 101: Matrix multiply.

- Compute $C \leftarrow C + A \cdot B$

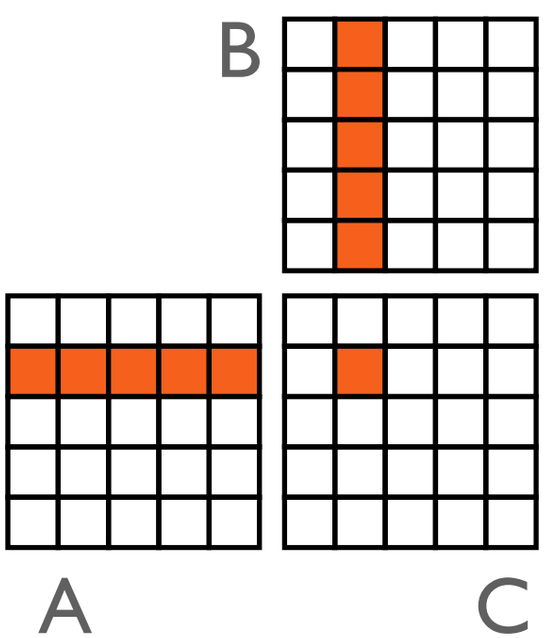


3-nested loops
(4 lines of code)

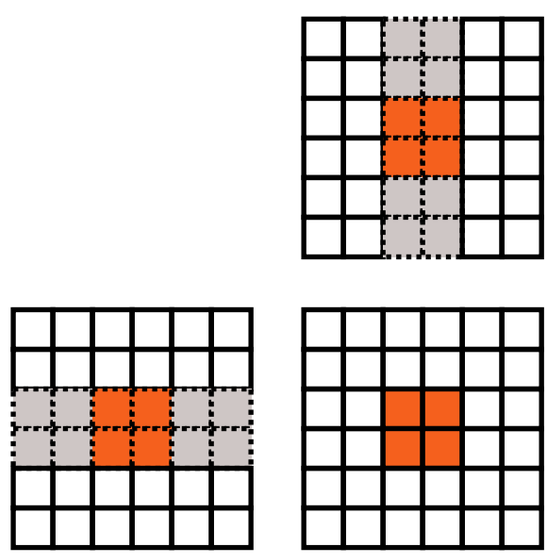
HPC 101: Cache-blocking is I/O optimal.

Computation:
$C \leftarrow A \cdot B$

Naïve



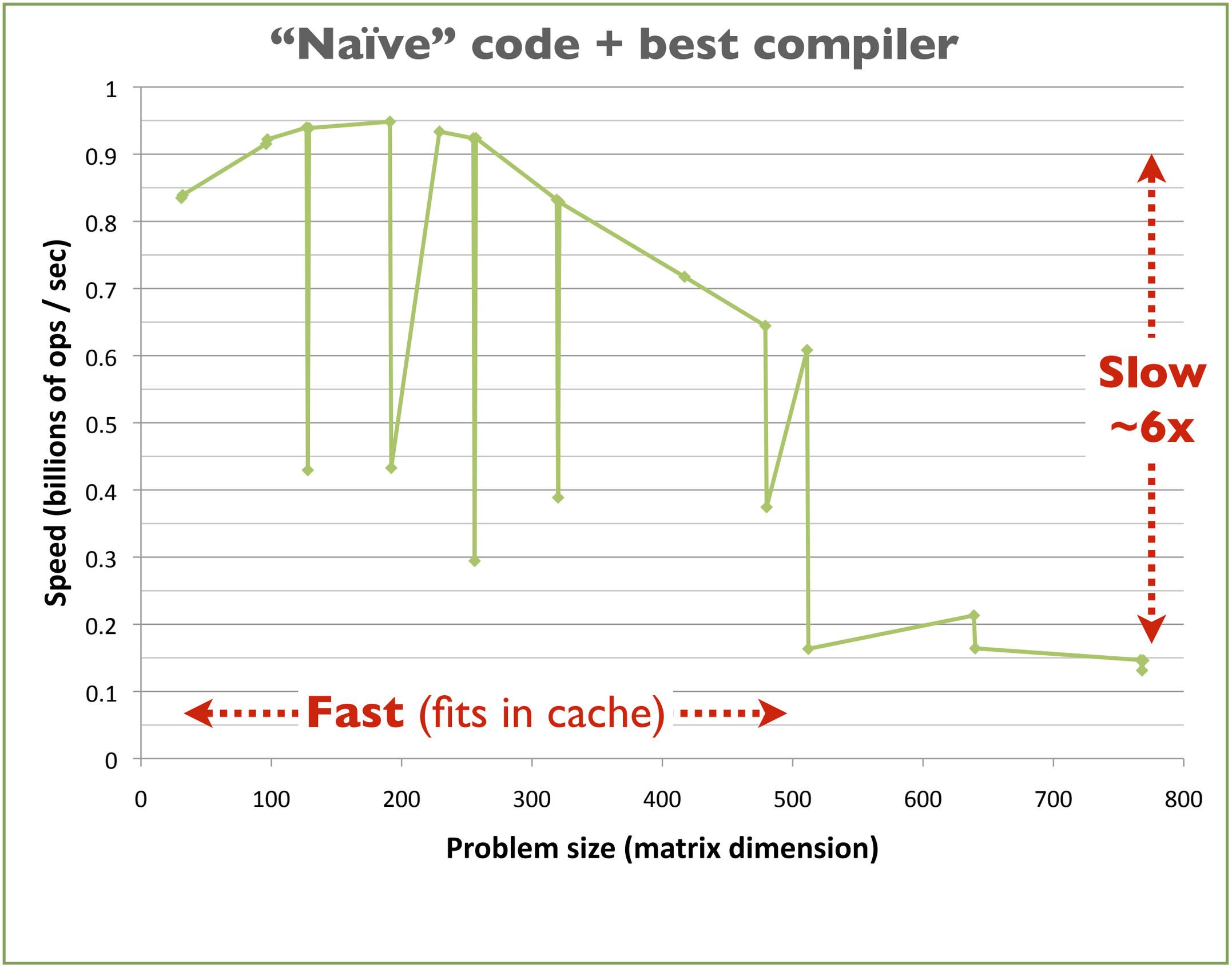
Blocked



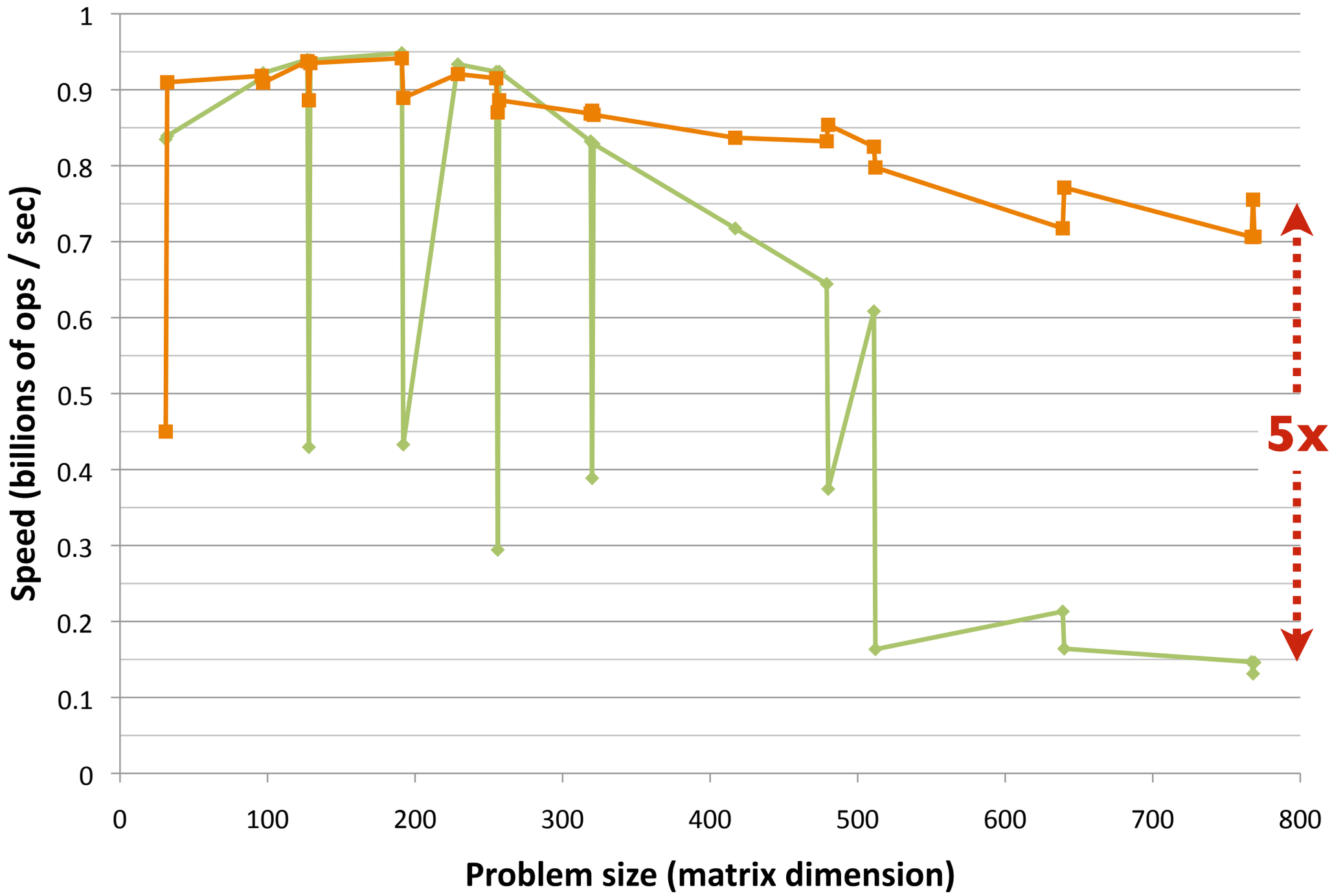
Theorem [Hong & Kung ('81)]:
A blocked algorithm minimizes slow memory transfers.

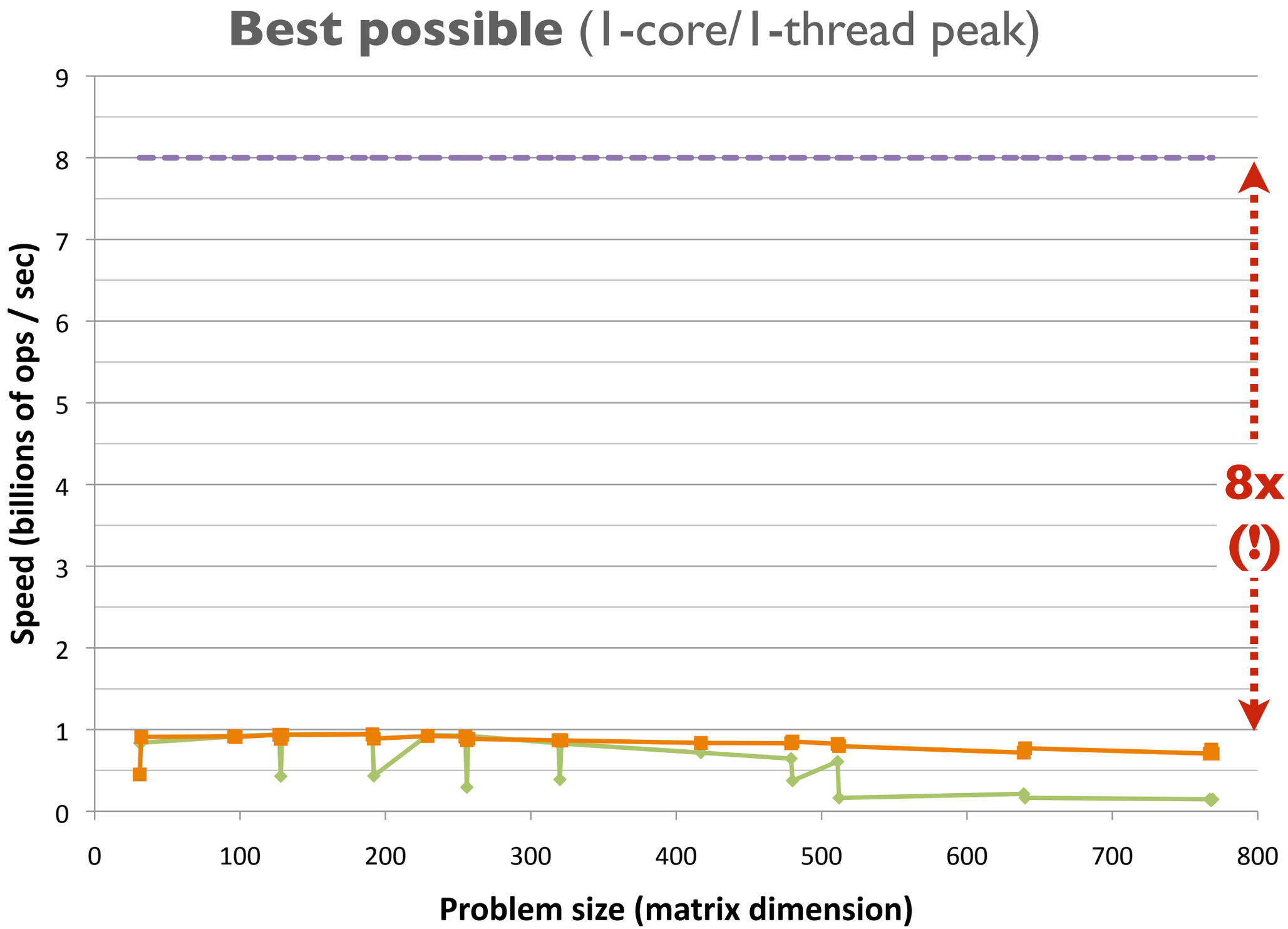
Tuning parameter: Block size

- That was theory.
  What happens in practice?

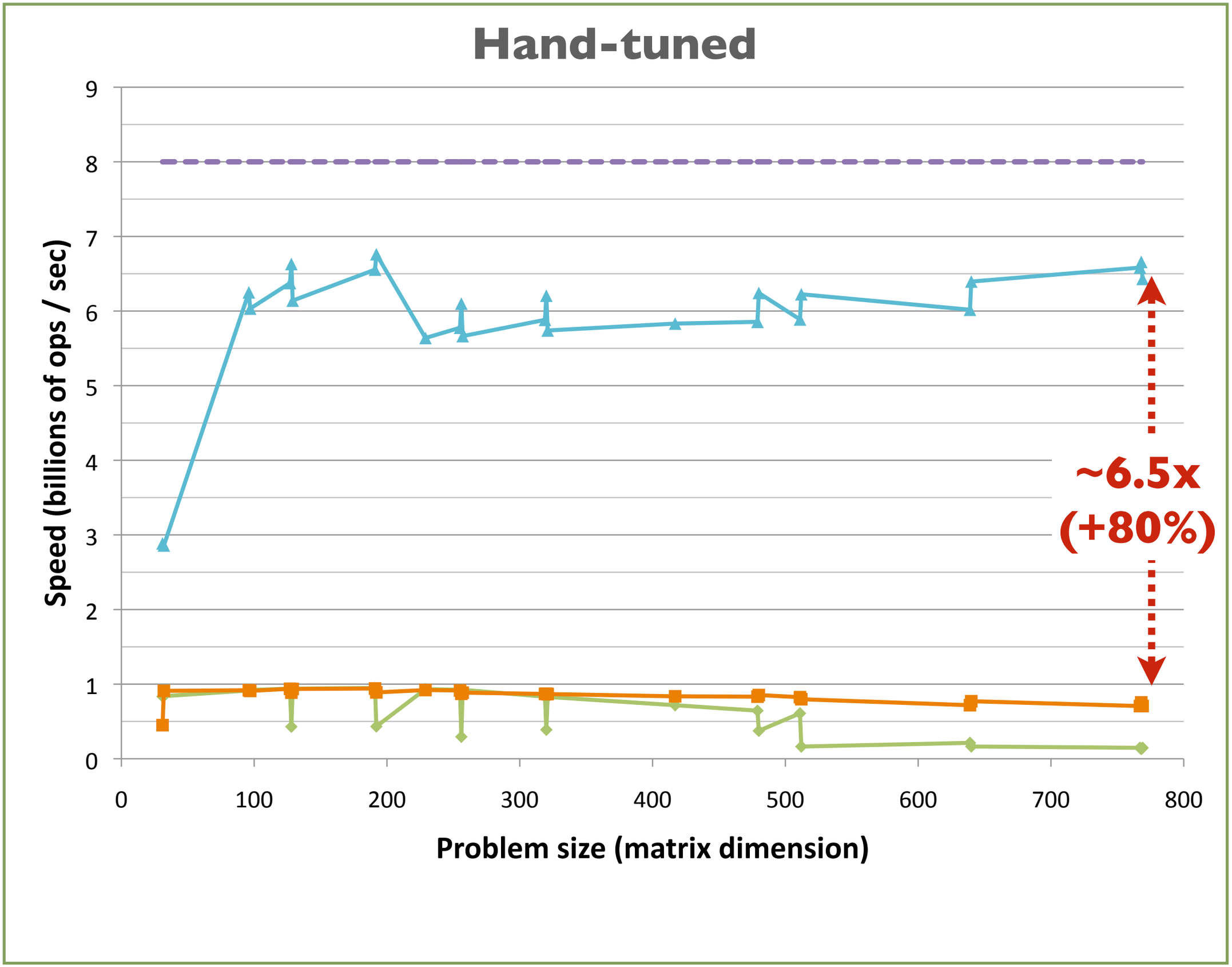

"Naïve" code + best compiler
Speed (billions of ops / sec)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
100
200
300
400
500
600
700
800
Problem size (matrix dimension)
Slow
~6x
Fast (fits in cache)

Blocked algorithm + best compiler

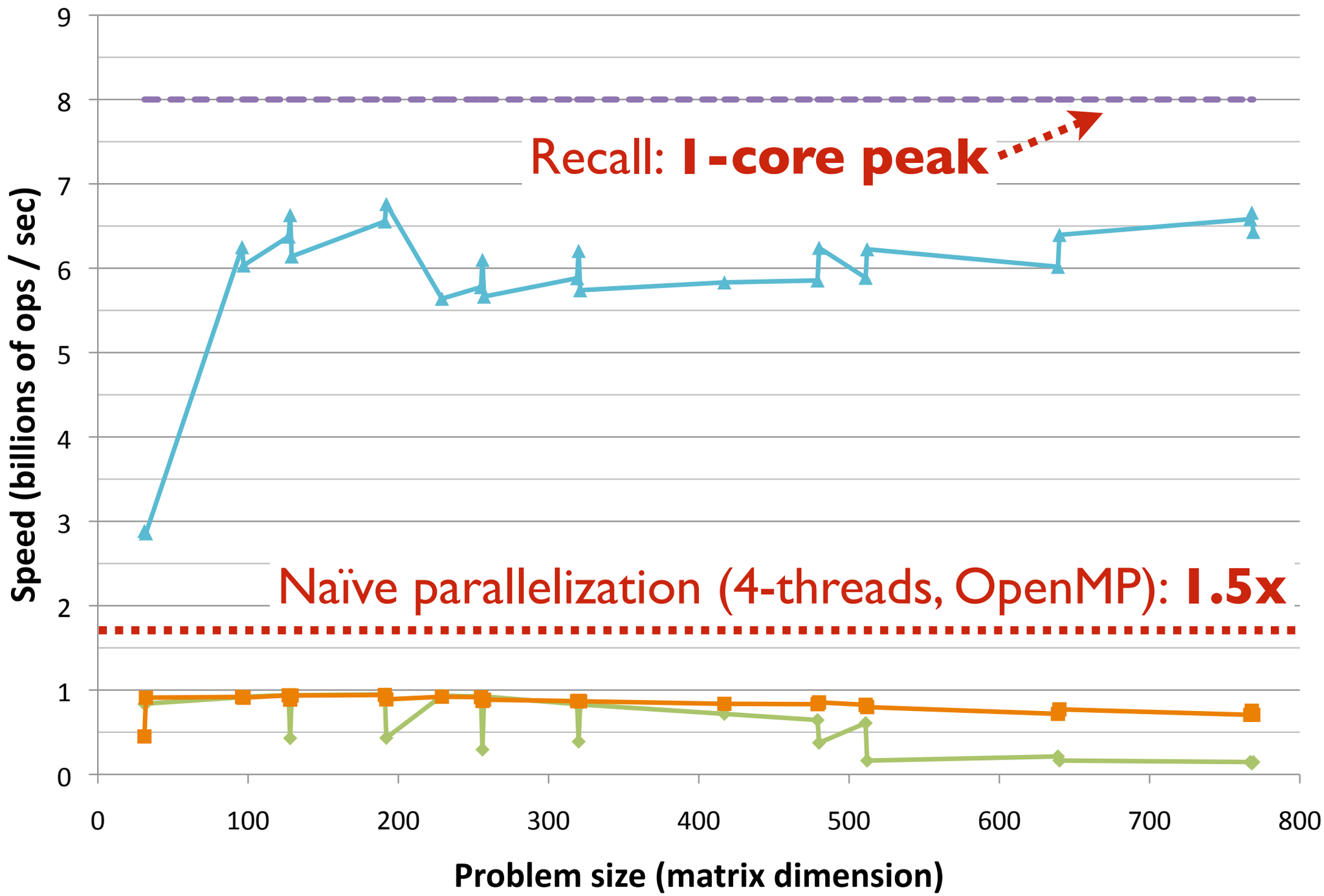




Peak = 8 Gflop/s on 1-core of 4-core 2 GHz AMD Opteron 8350


Hand-tuned
Speed (billions of ops / sec)
9
8
7
6
5
4
3
2
1
0
0
100
200
300
400
500
600
700
800
Problem size (matrix dimension)
~6.5x
(+80%)

Aside: Naïve parallelization on 4-core system



Goals for today & Friday

- Understand the theory's applicability and limits
  - "Prove" Hong & Kung theorem
  - What is relationship to data movement in a parallel system?
- Learn the basics of the memory hierarchy
  - Caches, TLB, memory
  - Ideas extend to disk systems

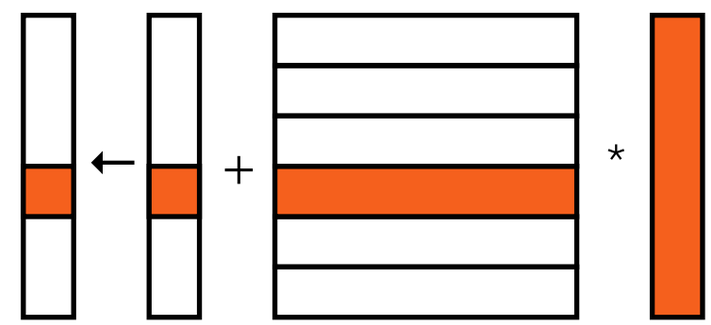
- A simple model of memory

A simple memory model

$$
\begin{aligned}
m &\equiv \text{No. words moved from slow to fast memory} \\
f &\equiv \text{No. of flops} \\
\alpha &\equiv \text{Time per slow memory op.} \\
\tau &\equiv \text{Time per flop} \\
q &\equiv \frac{f}{m} = \text{Flop-to-mop ratio} \quad \Leftarrow \textbf{Computational intensity} \\
T &= f \cdot \tau + m \cdot \alpha = f \cdot \tau \cdot \left(1 + \underbrace{\frac{\alpha}{\tau}}_{\textbf{Machine balance}} \cdot \frac{1}{q}\right)
\end{aligned}
$$

Example: Dense matrix-vector multiply

// Implements $y \leftarrow y + A \cdot x$



for $i \leftarrow 1$ **to** n **do**

  for $j \leftarrow 1$ **to** n **do**

    $y_i \leftarrow y_i + a_{ij} \cdot x_j$

Example: Dense matrix-vector multiply

// Implements $y \leftarrow y + A \cdot x$

// Read x (into fast memory)

// Read y

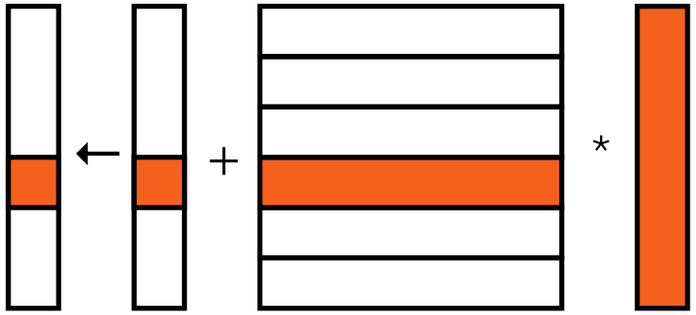
for $i \leftarrow 1$ **to** n **do**

  // Read $a_{i,\star}$

  for $j \leftarrow 1$ **to** n **do**

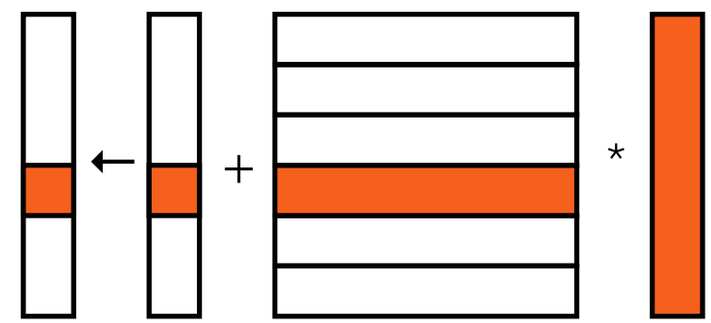
    $y_i \leftarrow y_i + a_{ij} \cdot x_j$

// Write y to slow memory



Example: Dense matrix-vector multiply

// Implements $y \leftarrow y + A \cdot x$
// Read x (into fast memory)
// Read y
for $i \leftarrow 1$ **to** n **do**
  // Read $a_{i,\star}$
  for $j \leftarrow 1$ **to** n **do**
    $y_i \leftarrow y_i + a_{ij} \cdot x_j$
// Write y to slow memory

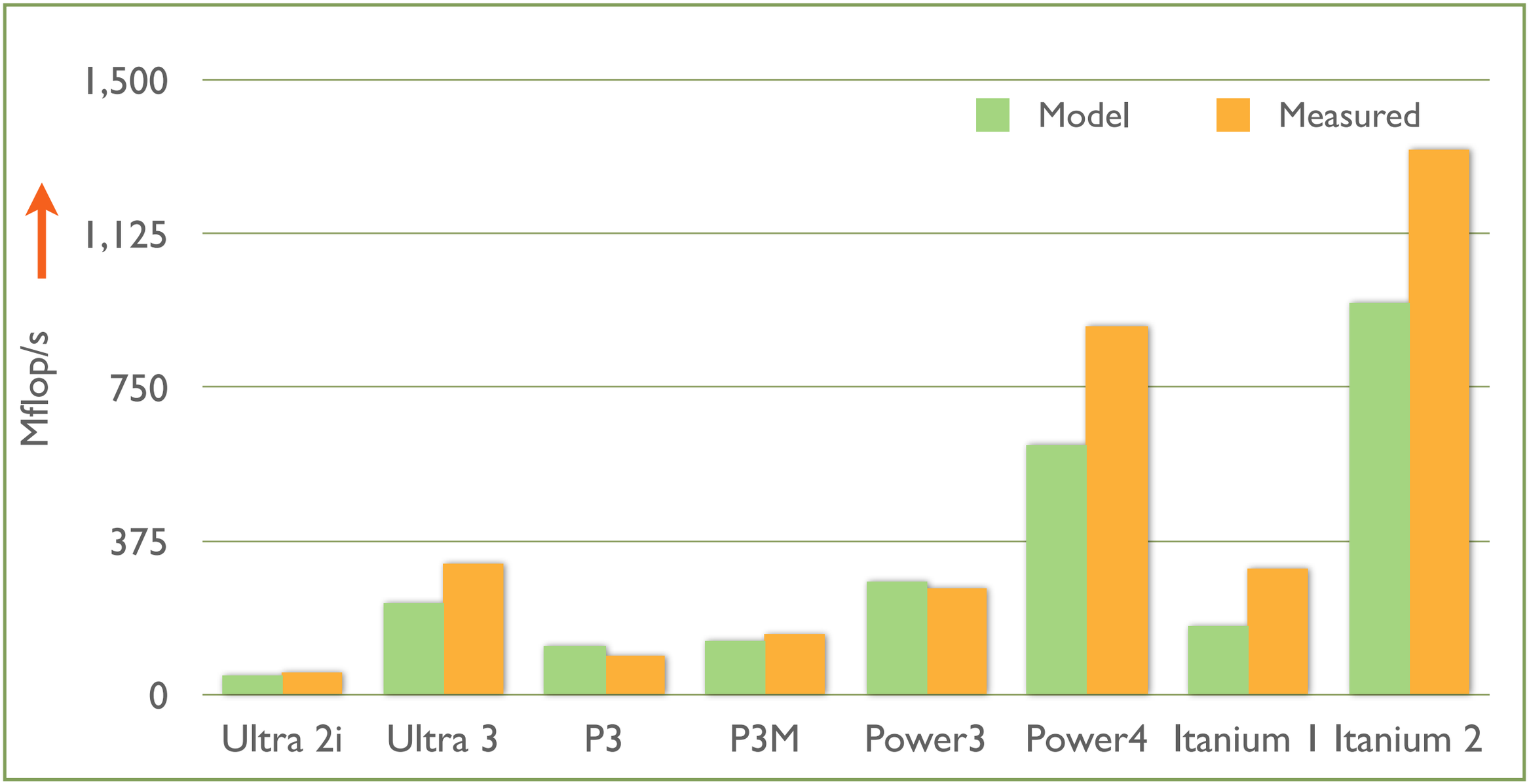


$$
\begin{aligned}
f &= 2n^2 \\
m &= 3n + n^2 \\
q &\approx 2 \\
&\Downarrow \\
\frac{T}{f \cdot \tau} &\approx 1 + \frac{\alpha}{\tau} \cdot \frac{1}{2}
\end{aligned}
$$

Simplifying assumptions

- Ignored flop/mop parallelism within processor → drop arithmetic term
- Assumed fast memory large enough to hold vectors
- Assumed no-cost fast memory access
- Memory latency is constant, charged per word
  - Ignored cache lines / block transfers
  - Ignored bandwidth

Predictive accuracy of this model



Naïve matrix-matrix multiply

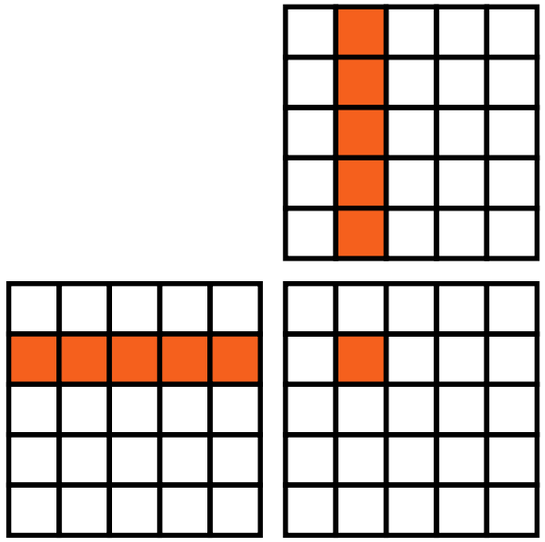
// Implements $C \leftarrow C + A \cdot B$

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

for $k \leftarrow 1$ **to** n **do**

$c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$



$$f = ?$$

$$m \geq ?$$

$$\frac{T}{f \cdot \tau} \geq ?$$

Naïve matrix-matrix multiply: Best case

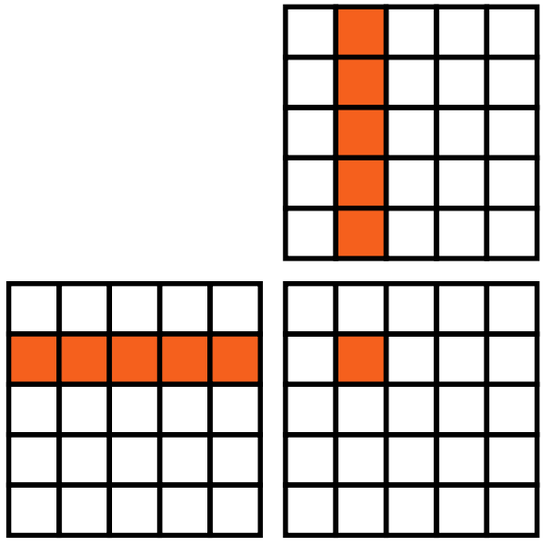
// Implements $C \leftarrow C + A \cdot B$

for $i \leftarrow 1$ **to** n **do**

for $j \leftarrow 1$ **to** n **do**

for $k \leftarrow 1$ **to** n **do**

$c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$



$$f = 2n^3$$

$$m \geq 4n^2$$

$$\frac{T}{f \cdot \tau} \geq 1 + \frac{\alpha}{\tau} \cdot \frac{2}{n}$$

Naïve matrix-matrix multiply

// Implements $C \leftarrow C + A \cdot B$

for $i \leftarrow 1$ **to** n **do**

// Read row $a_{i,\star}$

for $j \leftarrow 1$ **to** n **do**

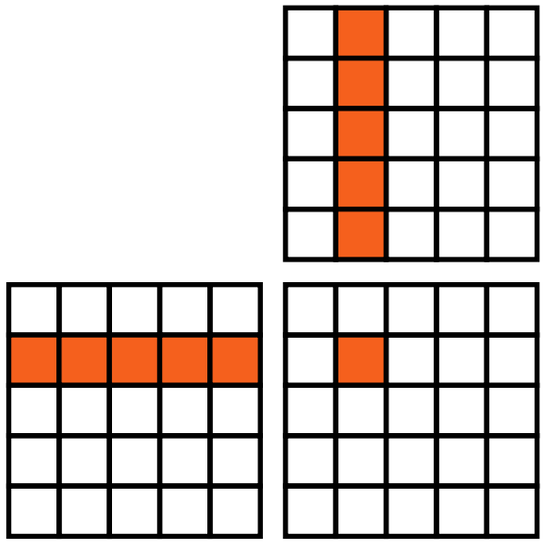
// Read col $b_{\star,j}$

// Read $c_{i,j}$

for $k \leftarrow 1$ **to** n **do**

$c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$

// Write c_{ij} to slow memory



$$f = 2n^3$$

$$m = n^3 + 3n^2$$

$$\frac{T}{f \cdot \tau} \approx 1 + \frac{\alpha}{\tau} \cdot \frac{1}{2}$$

Blocked (tiled) matrix multiply

// Let I, J, K = blocks of b indices

for $I \leftarrow$ index blocks 1 **to** $\frac{n}{b}$ **do**

  for $J \leftarrow$ index blocks 1 **to** $\frac{n}{b}$ **do**

    // Read block C_{IJ}

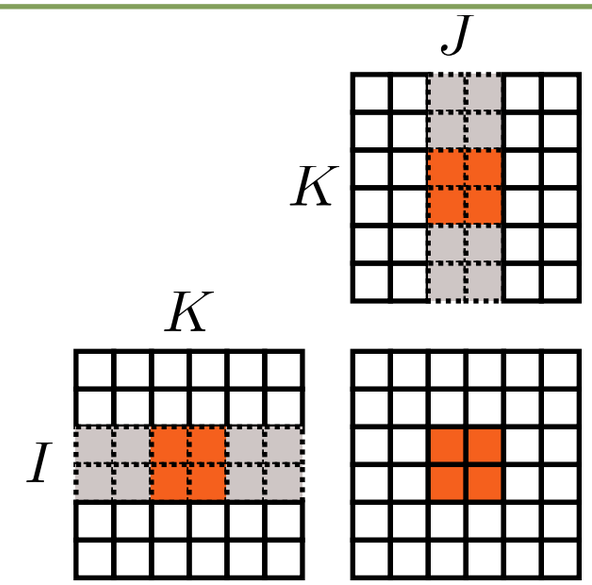
    for $K \leftarrow$ index blocks 1 **to** $\frac{n}{b}$ **do**

      // Read block A_{IK}

      // Read block B_{KJ}

      $C_{IJ} \leftarrow C_{IJ} + A_{IK} \cdot B_{KJ}$

    // Write C_{IJ} to slow memory



Blocked (tiled) matrix multiply

$$m \approx \frac{n^3}{b} \implies q \approx b$$

$$\frac{T}{f \cdot \tau} = 1 + \frac{\alpha}{\tau} \cdot \frac{1}{b}$$

Subject to what constraints?

Blocked (tiled) matrix multiply

$$m \approx \frac{2n^3}{b} \quad \Longrightarrow \quad q \approx b$$

$$3b^2 \leq Z \quad \Longrightarrow \quad b \leq \sqrt{\frac{Z}{3}}$$

$$\frac{T}{f \cdot \tau} = 1 + \frac{\alpha}{\tau} \cdot \frac{1}{b}$$

$$\geq 1 + \frac{\alpha}{\tau} \cdot \sqrt{\frac{3}{Z}}$$

Architectural implications

Arch.	≈ α / τ	Z
Ultra 2i	25	1.5 MB
Ultra 3	14	460 KB
Pentium 3	6.3	94 KB
P-3M	10	240 KB
Power3	8.8	180 KB
Power4	15	527 KB
Itanium 1	36	3.0 MB
Itanium 2	5.5	71 KB

"M" in bytes to 2 digits; assumes 8-byte (double-precision) words

$$1 + \frac{\alpha}{\tau} \cdot \frac{1}{q} < 1.1$$

$$\Longrightarrow Z \geq 300 \left(\frac{\alpha}{\tau}\right)^2$$

- I/O optimality

Optimal?

$$
\begin{aligned}
b &= O\left(\sqrt{Z}\right) \\
\Longrightarrow m &= O\left(\frac{n^3}{b}\right) = O\left(\frac{n^3}{\sqrt{Z}}\right)
\end{aligned}
$$

Bounding amount of I/O possible

Bounding amount of I/O possible

- Consider a schedule in phases of exactly Z transfers each (except last)
- *Definition*: $c(i,j)$ is **live** during phase p if ...
  - … for some k, we compute $a(i,k) * b(k,j)$;
  - *and* some partial sum of $c(i,j)$ is either in cache or moved to main memory
- At most 2*Z live $c(i,j)$ in phase p
- At most 2*Z distinct elements of A in cache during phase p; same for B
  - Either in cache at beginning or moved to cache during phase
  - Let A_p be set of elements in cache during phase p; same for B_p

How many multiplies in phase p?

- Let $S_{p,+}$ = set of rows of A with $Z^{1/2}$ or more elements in A_p
- Let $S_{p,-}$ = set of rows of A with fewer
- $|S_{p,+}| \leq 2*Z^{1/2}$
- Consider rows in $S_{p,+}$:
  - Operation "$a(i, :) \times B$" touches each element of B only once
  - So, no. of scalar multiplies $\leq |S_{p,+}| * (2*M) = 4*Z^{3/2}$
- For rows in $S_{p,-}$, consider that "$c(i,j)$ = row x col"
  - Thus, (# multiplies) $\leq$ (no. live) x (max row len) $\leq 2*Z^{3/2}$

Final bound on multiplies

$$
\begin{aligned}
\text{Total no. of multiplies} &= n^3 \\
\text{No. of multiplies per phase} &\leq 6Z^{\frac{3}{2}} \\
\text{No. of phases} &\geq \left\lceil \frac{n^3}{6Z^{\frac{3}{2}}} \right\rceil \\
\text{Total no. of words transferred} &\geq Z \cdot \left(\frac{n^3}{6Z^{\frac{3}{2}}} - 1 \right) \\
&= \frac{n^3}{6\sqrt{Z}} - Z
\end{aligned}
$$

Can we do better? No.

- *Theorem* [Hong & Kung (1981)]: Any schedule of conventional mat-mul must transfer $\Omega(n^3 / \sqrt{Z})$ words between slow and fast memory, where $Z < n^2 / 6$.
- We did intuitive proof by Toledo (1999)
- Historical note: Rutledge & Rubinstein (1951—52)
- So cached block matrix multiply is **asymptotically optimal**.

$$b = O\left(\sqrt{Z}\right) \implies m = O\left(\frac{n^3}{b}\right) = O\left(\frac{n^3}{\sqrt{Z}}\right)$$