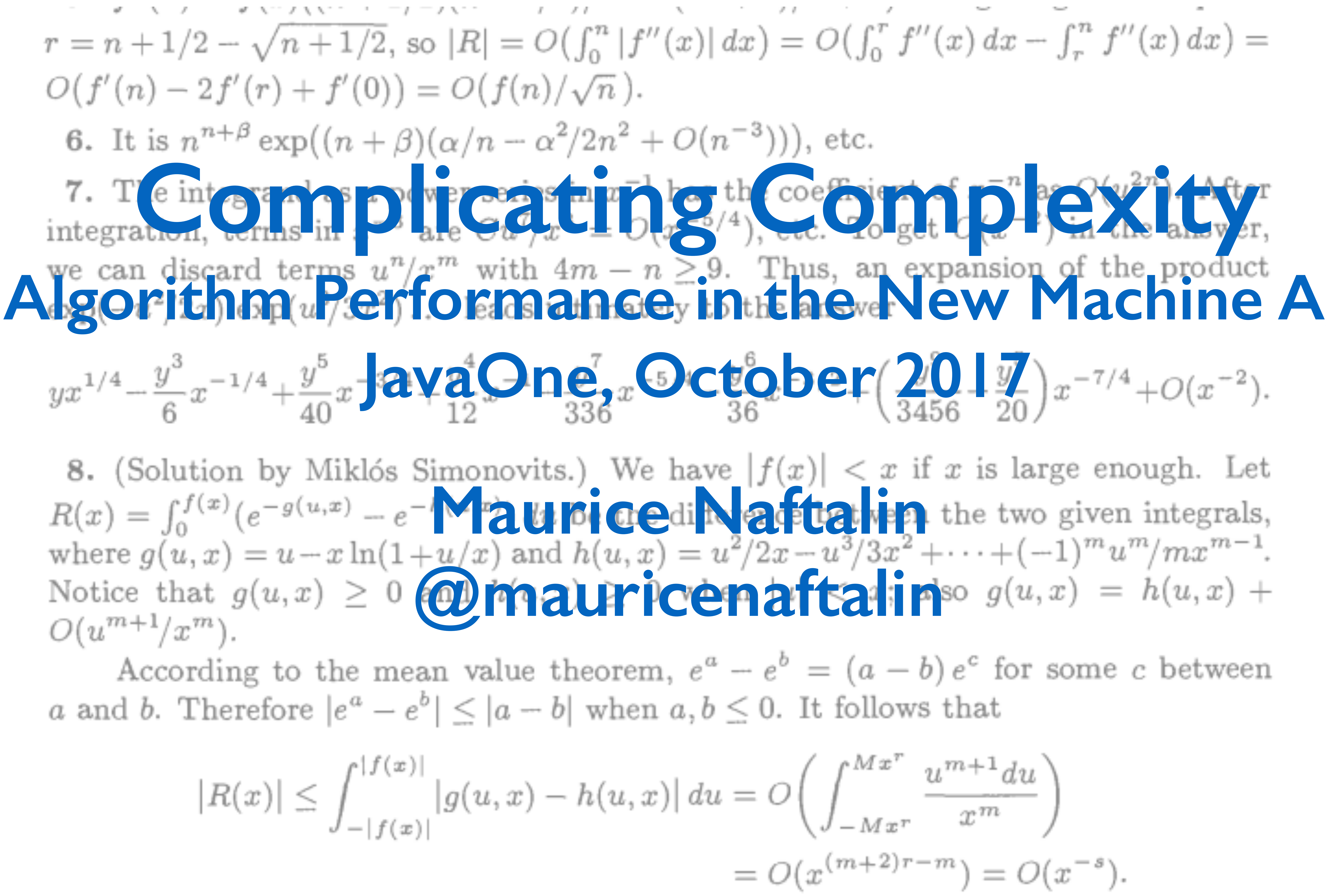
Complicating Complexity

Algorithm Performance in the New Machine Age

JavaOne, October 2017

Maurice Naftalin

@mauricenaftalin



$r = n + 1/2 - \sqrt{n+1/2}$, so $|R| = O(\int_0^n |f''(x)|\,dx) = O(\int_0^r f''(x)\,dx - \int_r^n f''(x)\,dx) = O(f'(n) - 2f'(r) + f'(0)) = O(f(n)/\sqrt{n}\,)$.

6. It is $n^{n+\beta}\exp((n+\beta)(\alpha/n - \alpha^2/2n^2 + O(n^{-3})))$, etc.

7. T[illegible]e int[illegible] th[illegible] coe[illegible] $O(u^{2n})$. After integration, terms in [illegible] are [illegible] $= O(x^{-5/4})$, etc. To get $O(x^{-2})$ in the answer, we can discard terms u^n/x^m with $4m - n \geq 9$. Thus, an expansion of the product [illegible] leads ultimately to the answer

$$yx^{1/4} - \frac{y^3}{6}x^{-1/4} + \frac{y^5}{40}x^{-3/4} + \frac{y^4}{12}x^{-1} - \frac{y^7}{336}x^{-5/4} - \frac{y^6}{36}x^{-3/2} + \left(\frac{y^9}{3456} + \frac{y^5}{20}\right)x^{-7/4} + O(x^{-2}).$$

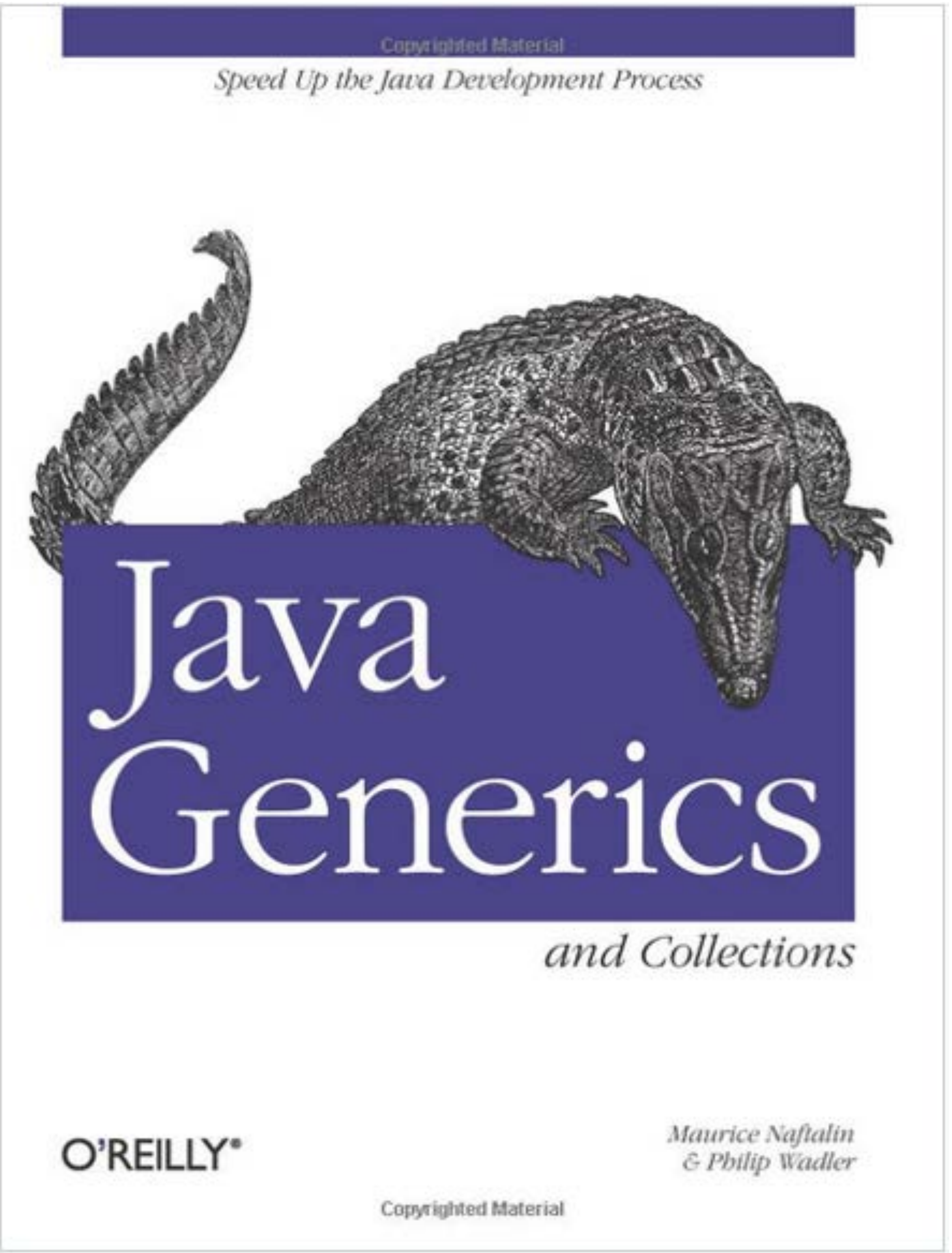
8. (Solution by Miklós Simonovits.) We have $|f(x)| < x$ if x is large enough. Let $R(x) = \int_0^{f(x)}(e^{-g(u,x)} - e^{-h(u,x)})$ [illegible] the two given integrals, where $g(u,x) = u - x\ln(1+u/x)$ and $h(u,x) = u^2/2x - u^3/3x^2 + \cdots + (-1)^m u^m/mx^{m-1}$. Notice that $g(u,x) \geq 0$ and [illegible] also $g(u,x) = h(u,x) + O(u^{m+1}/x^m)$.

According to the mean value theorem, $e^a - e^b = (a-b)\,e^c$ for some c between a and b. Therefore $|e^a - e^b| \leq |a - b|$ when $a, b \leq 0$. It follows that

$$|R(x)| \leq \int_{-|f(x)|}^{|f(x)|} |g(u,x) - h(u,x)|\,du = O\left(\int_{-Mx^r}^{Mx^r} \frac{u^{m+1}du}{x^m}\right)$$
$$= O(x^{(m+2)r-m}) = O(x^{-s}).$$

Maurice Naftalin

Java 5



Speed Up the Java Development Process

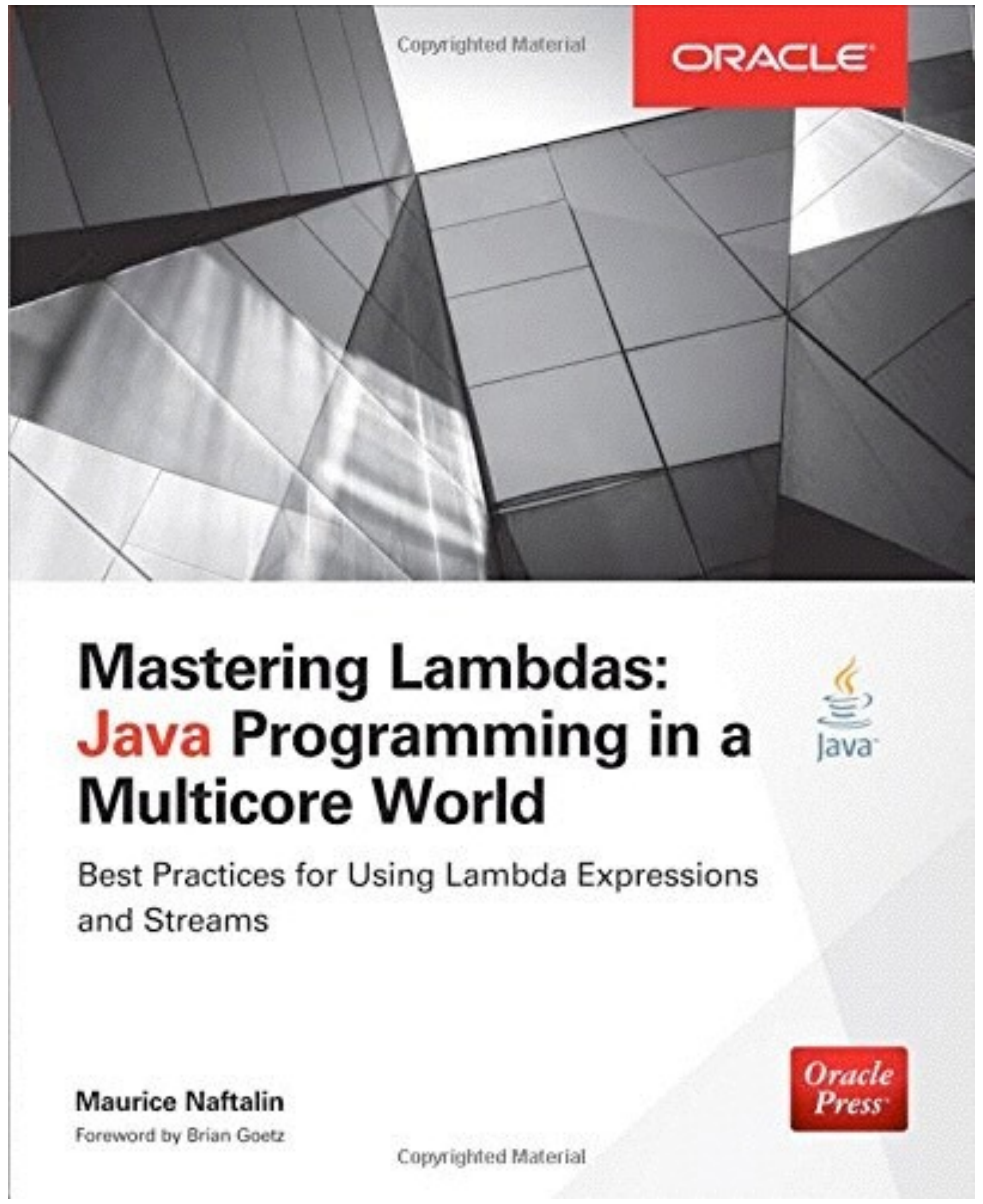
Java Generics

and Collections

O'REILLY®

Maurice Naftalin & Philip Wadler

Java 8



ORACLE

Mastering Lambdas: Java Programming in a Multicore World

Best Practices for Using Lambda Expressions and Streams

Maurice Naftalin

Foreword by Brian Goetz

ORACLE Developer Champion

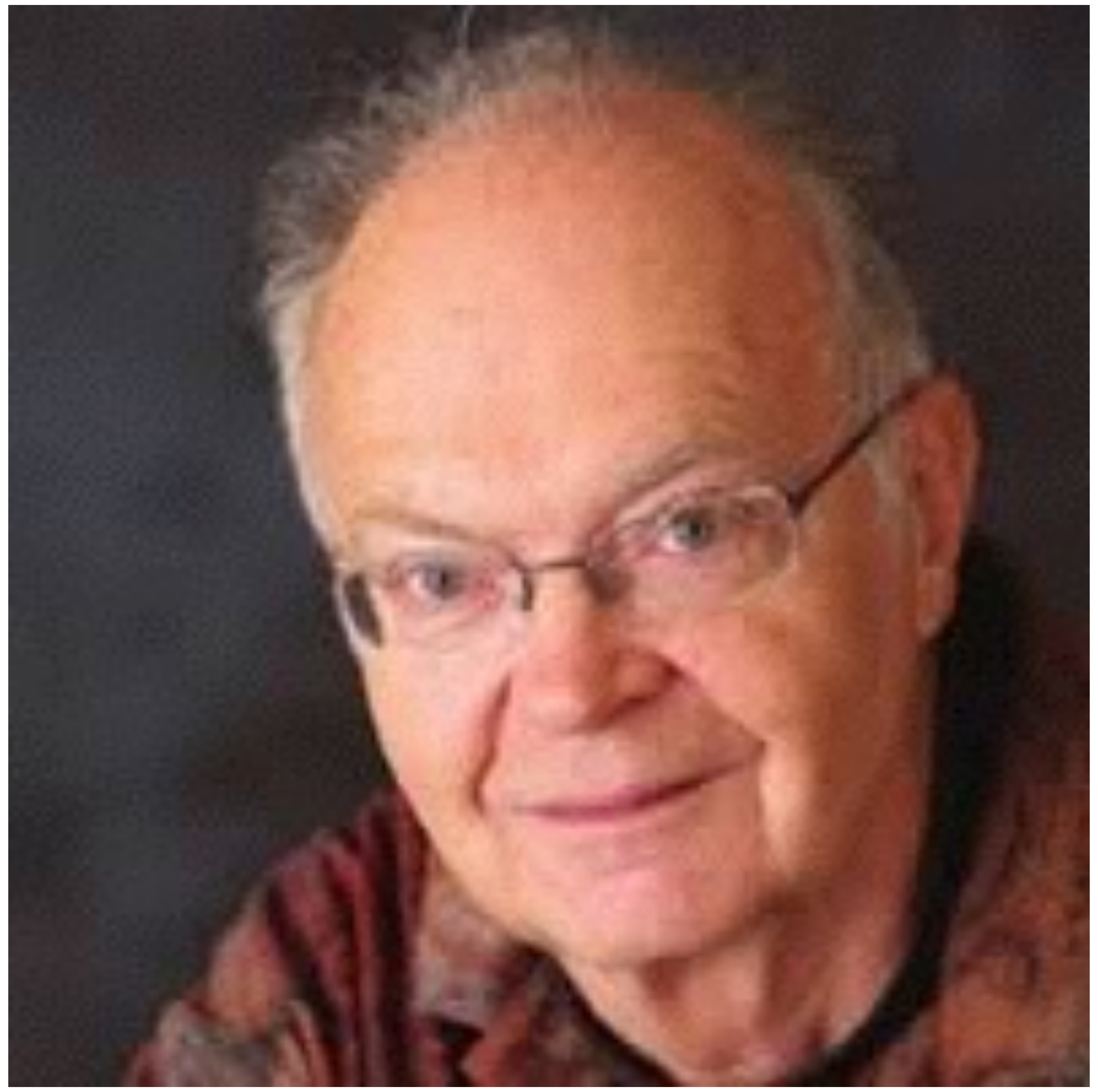
2013 2014 2015

Not Your Father's Complexity

- Original title for this talk
- Mental model of program performance
  - Number of instructions executed
- Why we need to change this, and how

Who *Is* Your Father?

…professionally speaking!



Don Knuth (1938 –), Creator of TeX,
Turing Award laureate,…

Author of *The Art of Computer Programming*

THE ART OF COMPUTER PROGRAMMING
VOLUME 4 PRE-FASCICLE 5A
A (PARTIAL) DRAFT OF SECTION 7.2.2.2: SATISFIABILITY
THE ART OF COMPUTER PROGRAMMING
MATHEMATICAL PRELIMINARIES REDUX
KNUTH
The Art of Computer Programming
MIX

What He Really Thinks

“Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered.

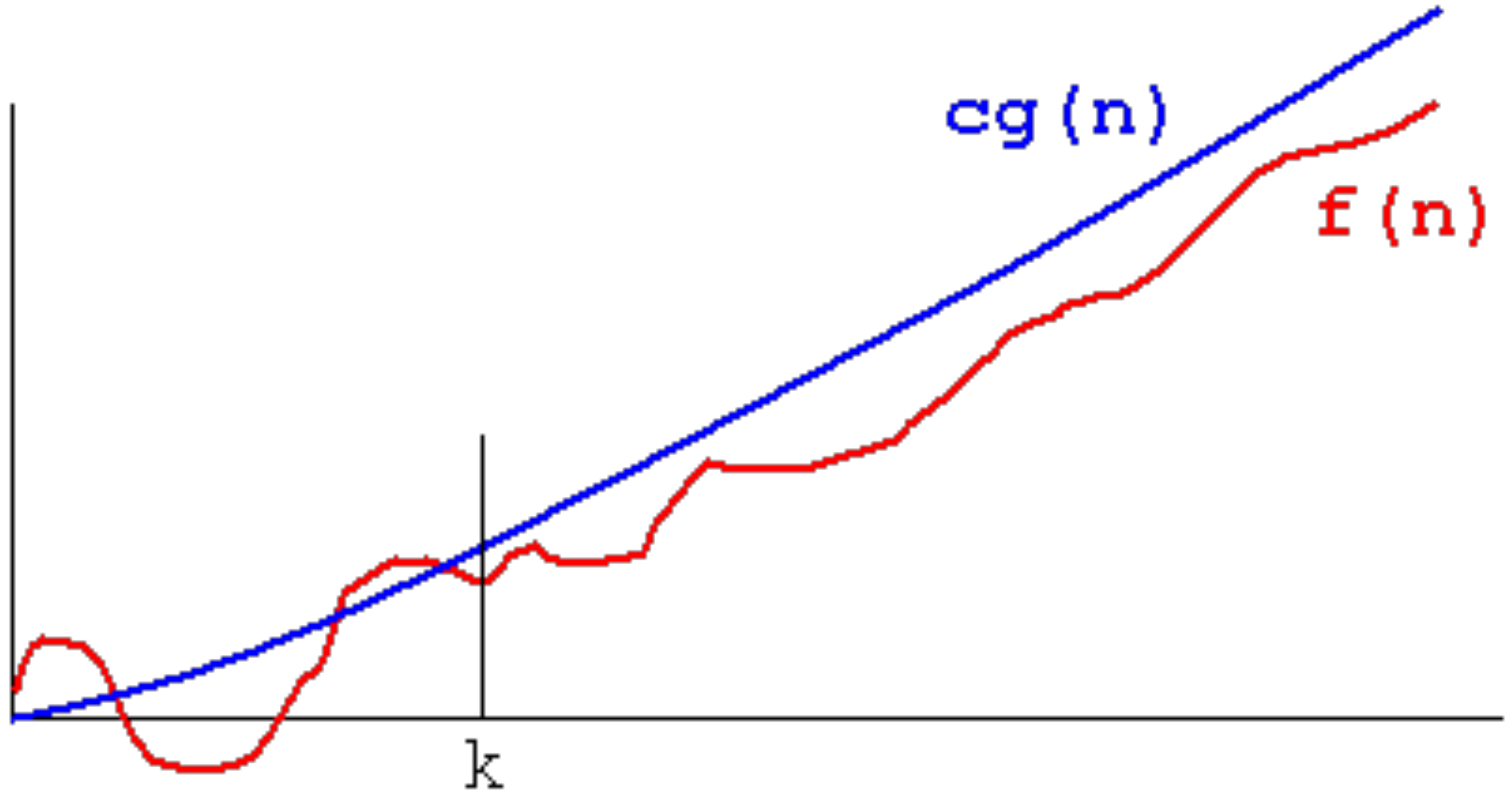
“We should forget about small efficiencies, say about 97% of the time:

premature optimization is the root of all evil”

What is Computational Complexity?

- an algorithm with time complexity $O(g(n))$ will complete in less than $c * g(n)$ steps, for some c and sufficiently large n

- e.g. Knuth, on multiple list insertion sort: execution time is
  $$3.5N^2 + 24.5N + 4M + 2$$
  - but we only care about the N^2 term!



Why Bother?

It’s the traditional way of evaluating algorithms!

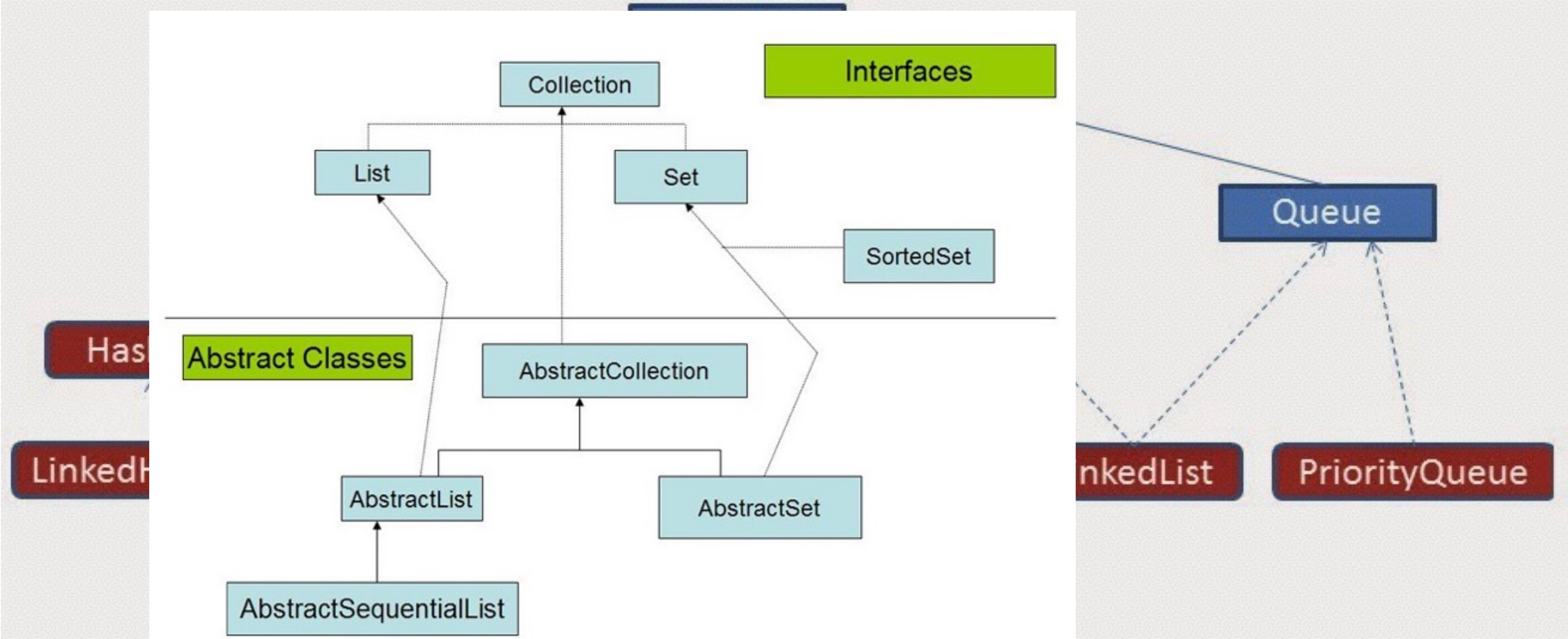
Complexity	Effect of doubling N
$O(1)$	Unchanged
$O(\log N)$	Increased by a constant
$O(N)$	Doubled
$O(N \log N)$	Doubled + an amount proportional to N
$O(N^2)$	Increased fourfold
…	…
…	…
…	…

Who *Is* Your Father?

Another Candidate:

Joshua Bloch (1961 –), Author of *Effective Java*

Author of the Java Collections Framework



What Josh Told Me…

- the Java Collections Framework is *interface-based*
- choose the interface (`Set, List, Queue`) that meets your requirement
- then choose the implementation with best performance for your usage scenario
- for example: which `List` implementation?
  - `ArrayList, LinkedList, CopyOnWriteArrayList?`

Which List Implementation?

	get()	add()	remove(0)
ArrayList	O(1)	O(1)	O(N)
LinkedList	O(N)	O(1)	O(1)
COWArrayList	O(1)	O(N)	O(N)

Did this ever make sense?

- Yes, on these assumptions:
  - can ignore constant factors
  - all instructions have the same duration
  - memory doesn’t matter
  - instruction execution dominates performance

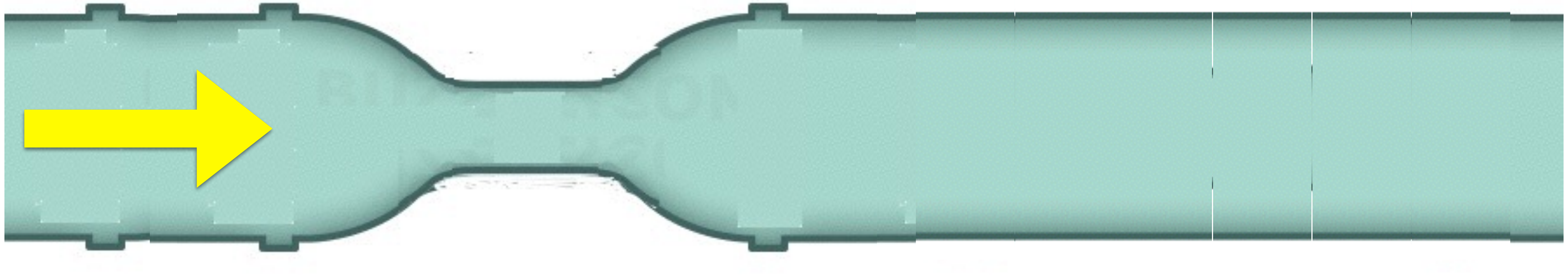
Was Complexity Study Ever Worth It?

Of course it was!

But instruction execution is only one bottleneck. Many others:

- Disk/Network
- Garbage Collection
- Resource Contention

and more…



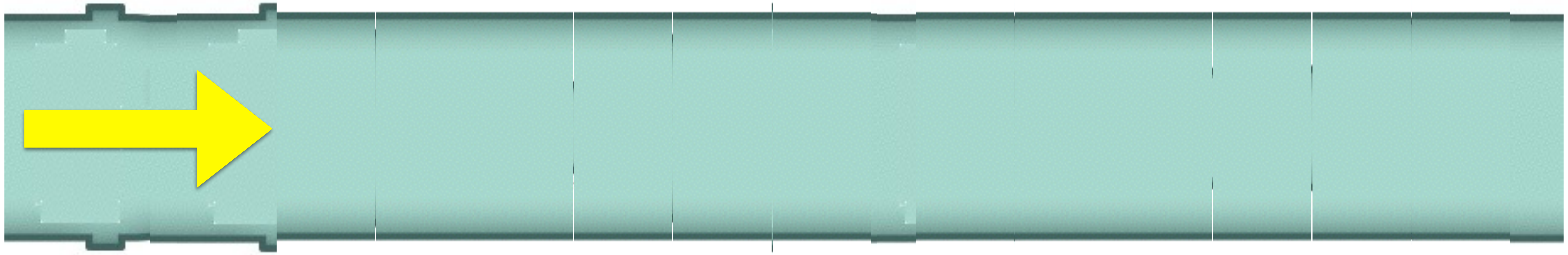
Was Complexity Study Ever Worth It?

Of course it was!

But instruction execution is only one bottleneck. Many others:

- Disk/Network
- Garbage Collection
- Resource Contention

and more…



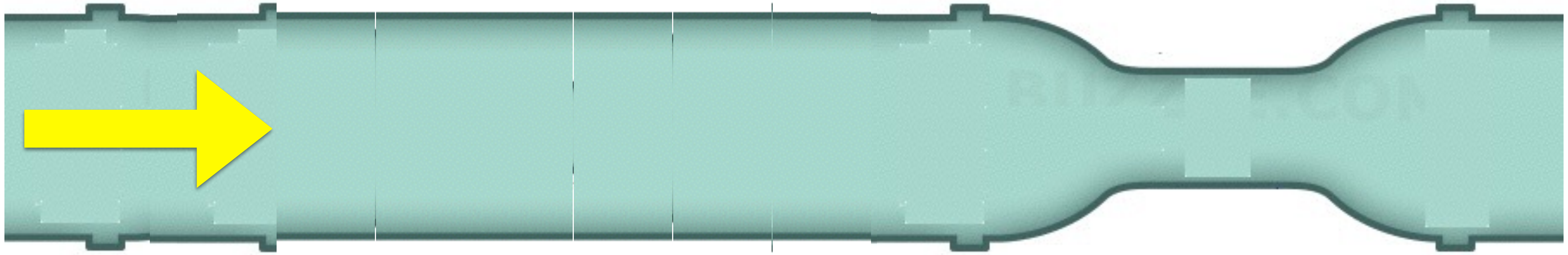
Was Complexity Study Ever Worth It?

Of course it was!

But instruction execution is only one bottleneck. Many others:

- Disk/Network
- Garbage Collection
- Resource Contention

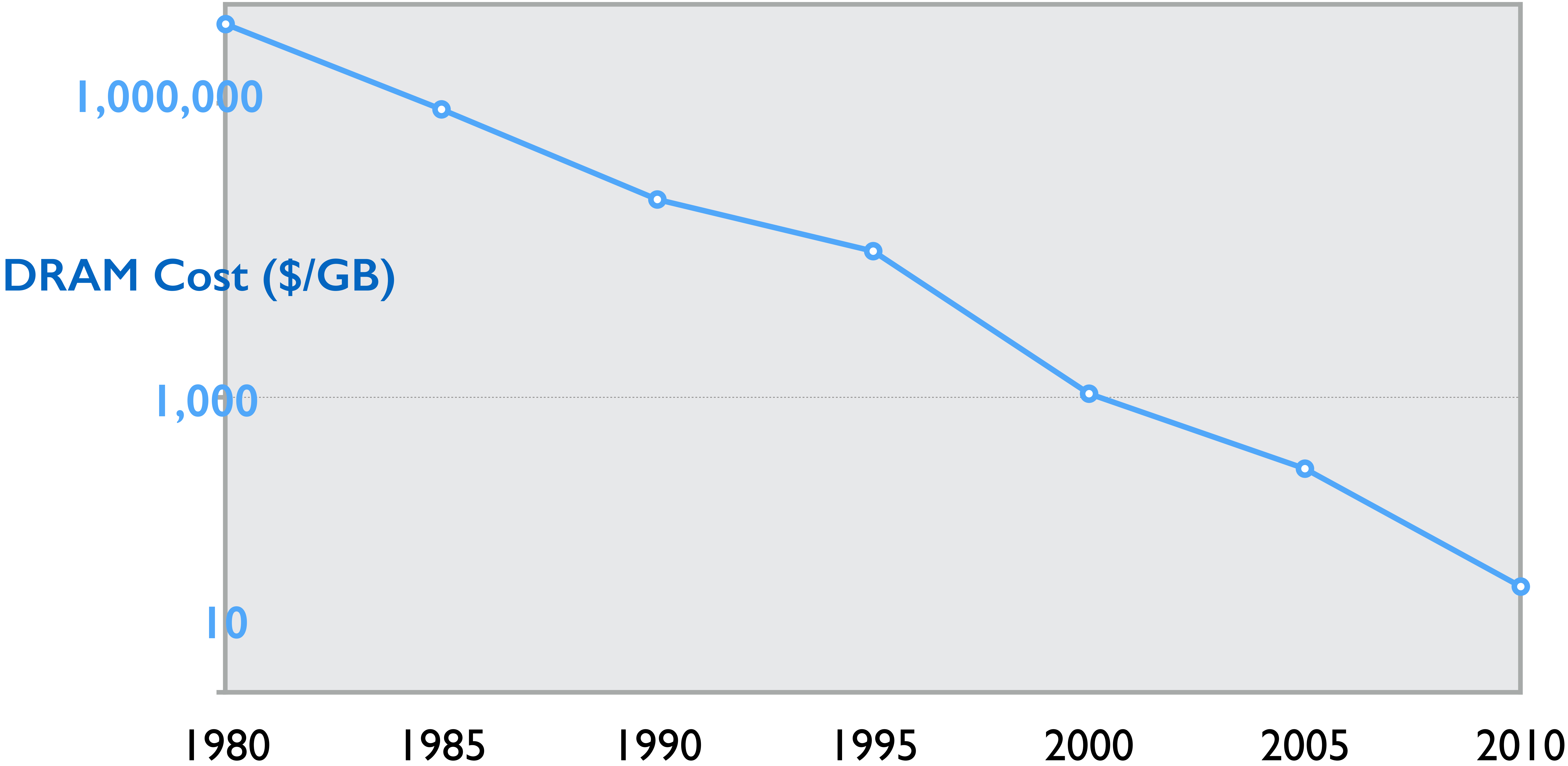
and more…



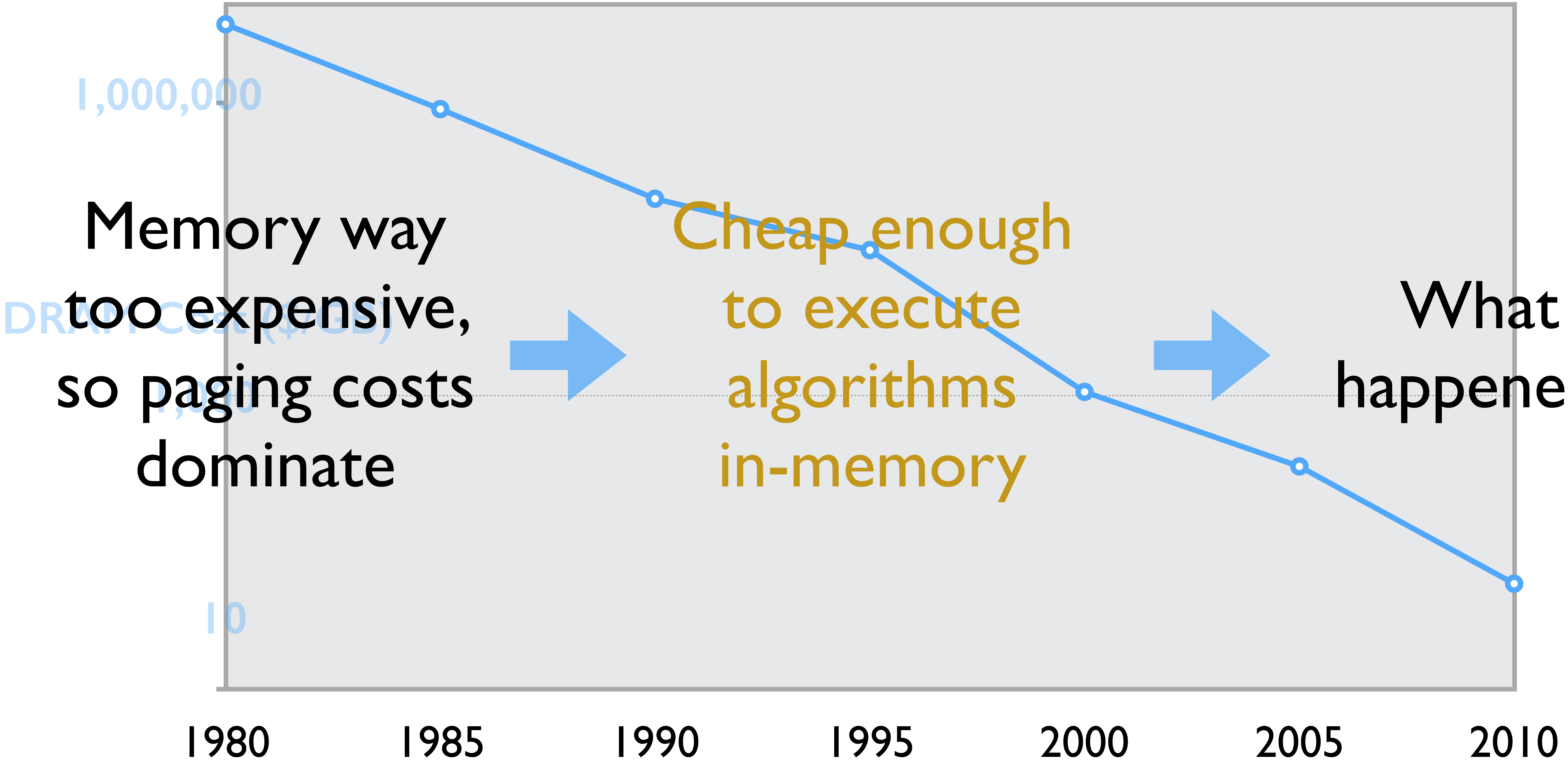
pre-1980

pre-1980

The Golden Age of Complexity



The Golden Age of Complexity

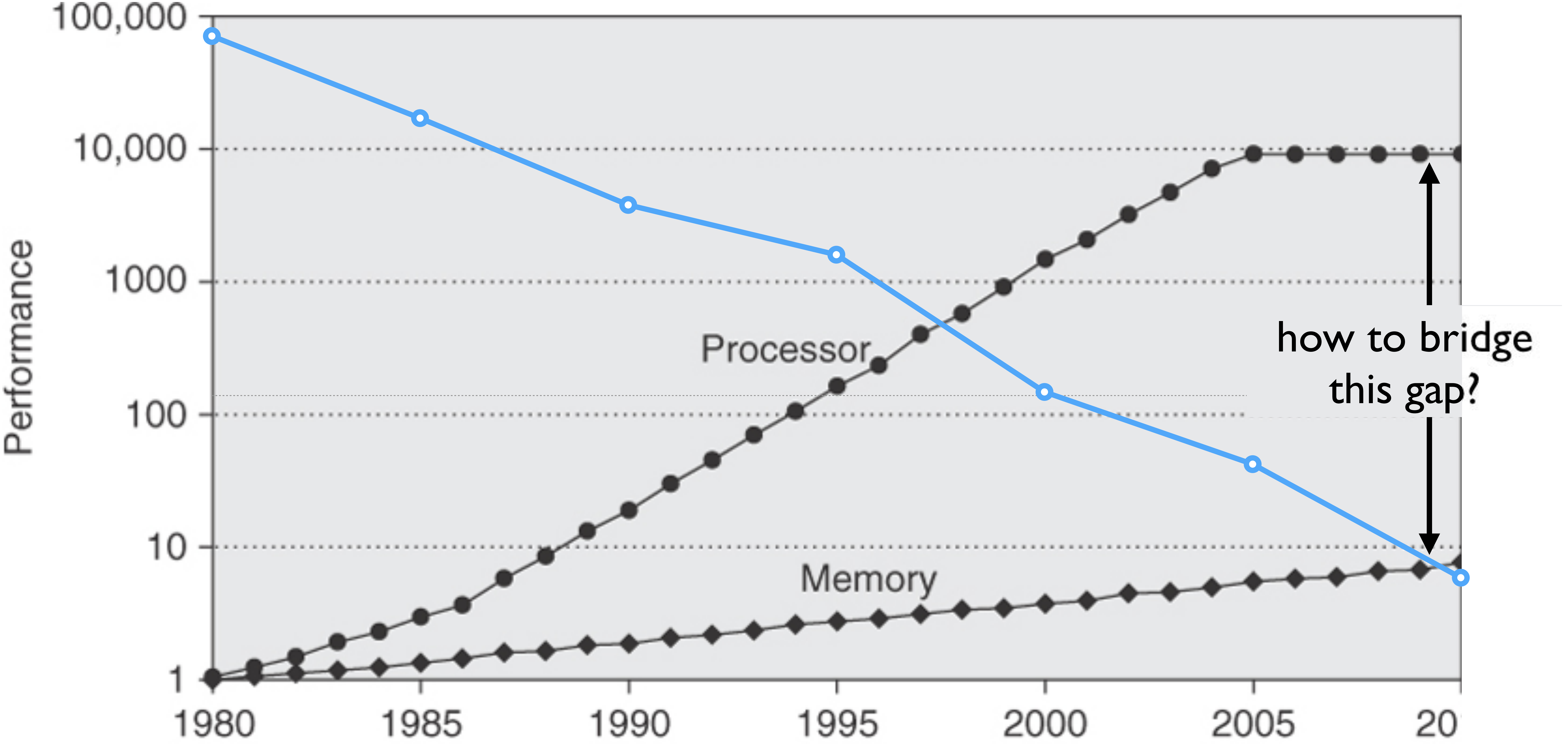


Memory way too expensive, so paging costs dominate

Cheap enough to execute algorithms in-memory

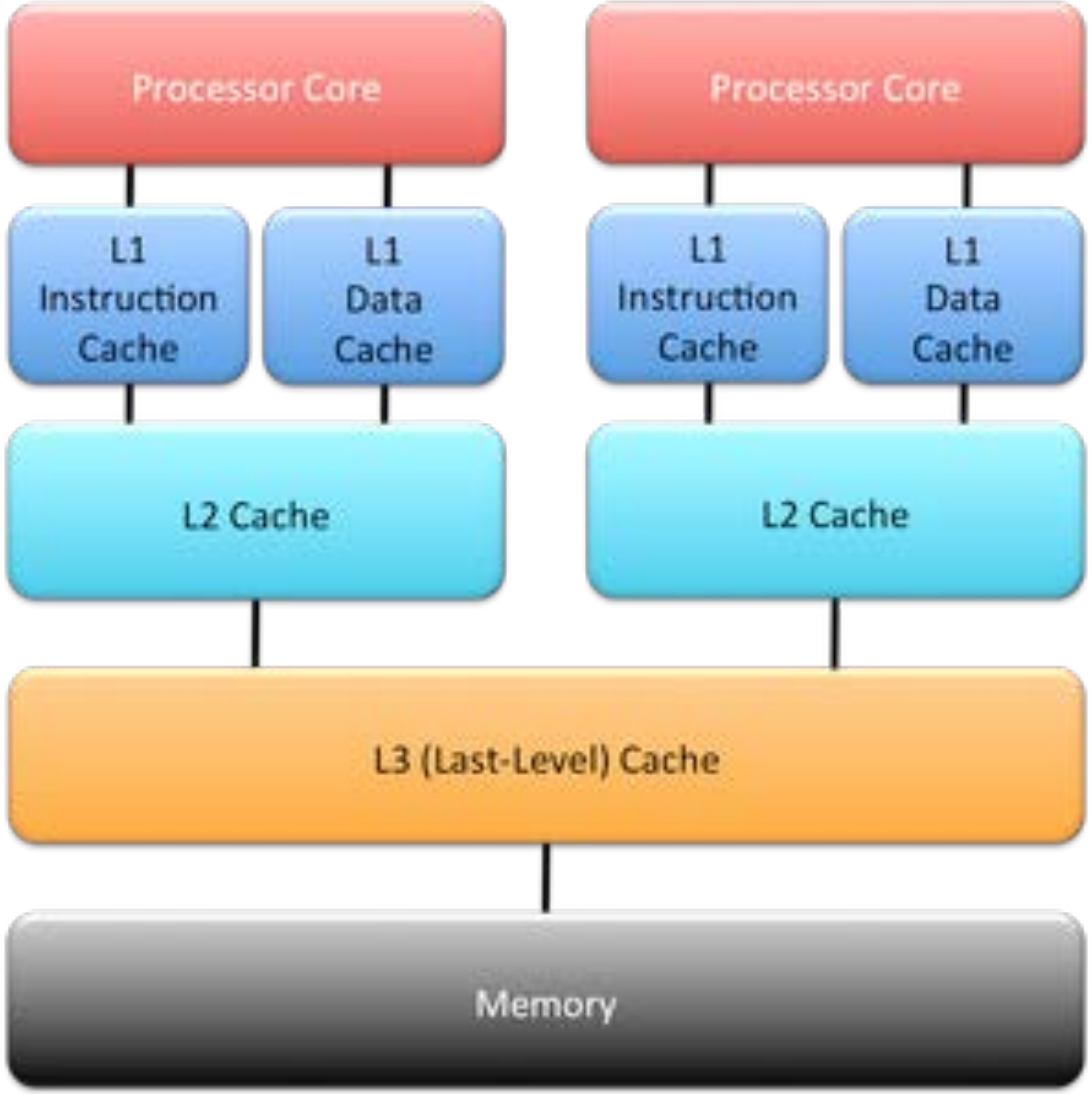
What happened?

The Golden Age of Complexity



how to bridge
this gap?

Keeping the Cores Running Today



The Memory Hierarchy

PROCESSOR
CPU
PROCESSOR REGISTER
SUPER FAST
SUPER EXPENSIVE
TINY CAPACITY
CPU CACHE
LEVEL 1 (L1) CACHE
LEVEL 2 (L2) CACHE
LEVEL 3 (L3) CACHE
FASTER
EXPENSIVE
SMALL CAPACITY
SD-RAM,
DDR-SDRAM, ...
PHYSICAL MEMORY
RANDOM ACCESS MEMORY (RAM)
FAST
PRICED REASONABLY
AVERAGE CAPACITY
SOLID STATE DRIVES
SOLID STATE MEMORY
NON-VOLATILE FLASH-BASED MEMORY
AVERAGE SPEED
PRICED REASONABLY
AVERAGE CAPACITY
MECHANICAL HARD DRIVES
VIRTUAL MEMORY
FILE-BASED MEMORY
SLOW
CHEAP
LARGE CAPCITY

<< Advantages >>

Speed Power Cost

The Memory Hierarchy

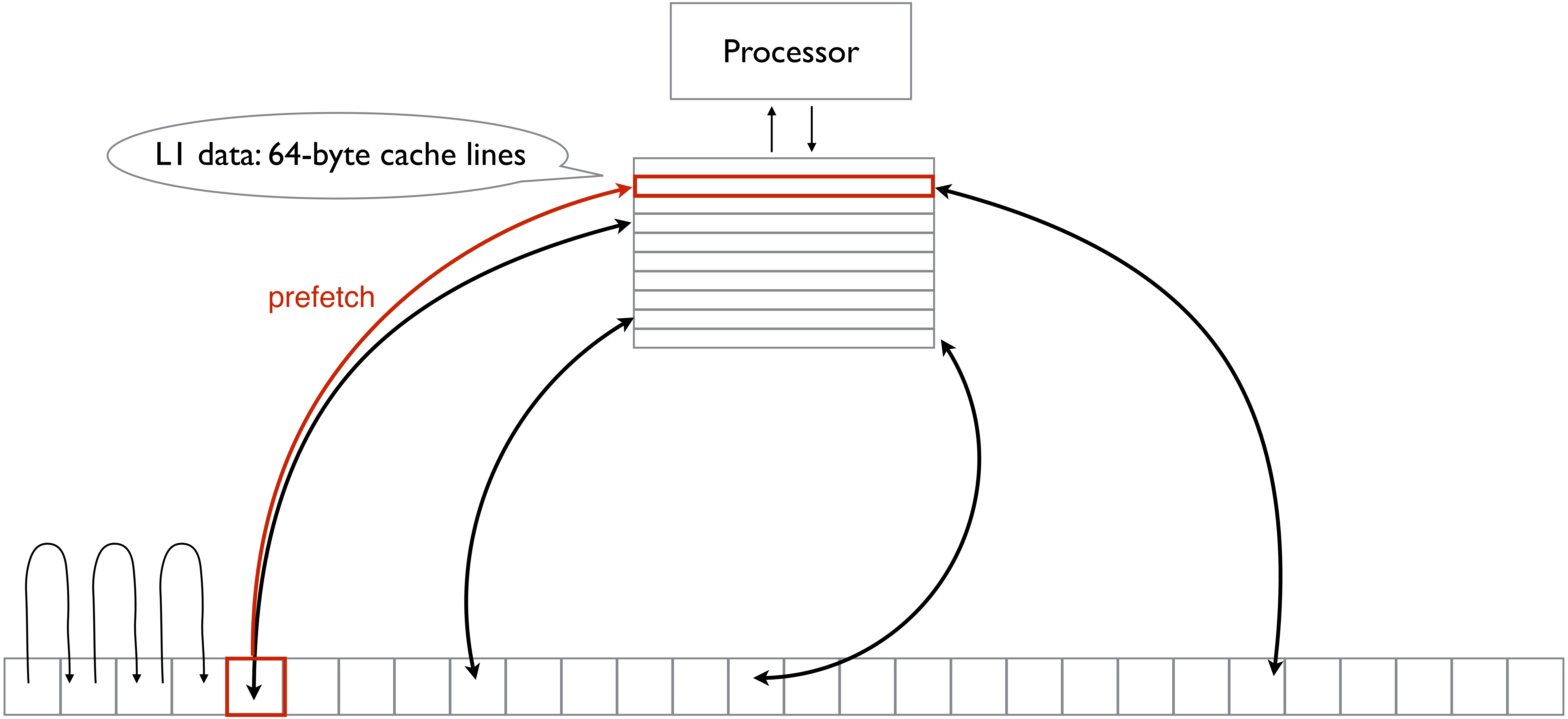
Cache Effects Often Dominate

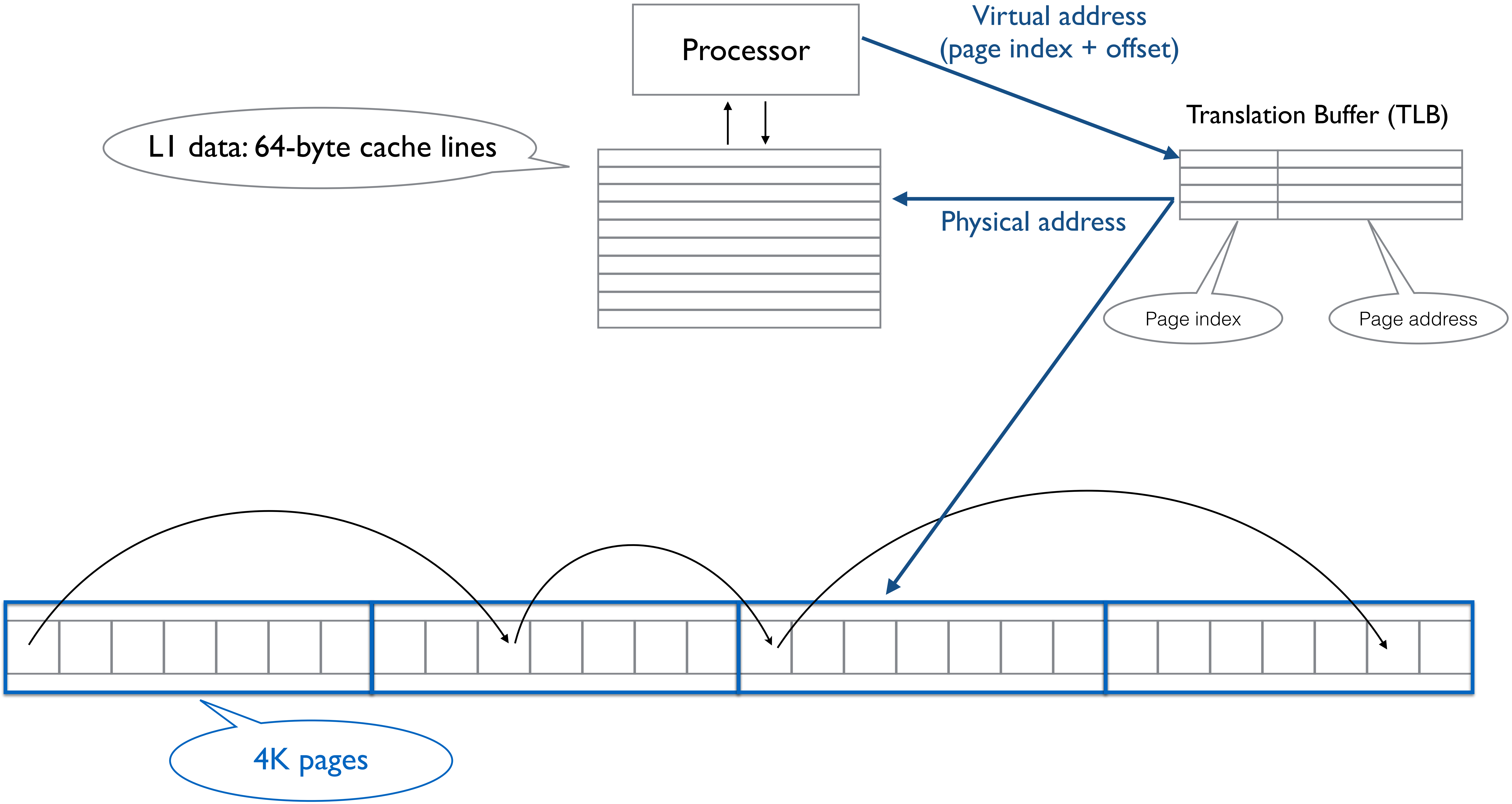
- Main memory retrieval costs ~100x L1 access
    - 2-300x register access

- Typical programs have 95% hit rate
    - it's the other 5% that hurts

- Why cache misses? Two possible reasons (many others):
    - insufficient capacity
    - failure of prefetching
        - unpredictable data access patterns!

Stride Prefetching




Processor
Virtual address
(page index + offset)
Translation Buffer (TLB)
L1 data: 64-byte cache lines
Physical address
Page index
Page address
4K pages

How Does Caching Play With Complexity?

Sample case: traversing a list

– *O*(n), obviously

First issue: data size. Let's compare:

- `LinkedList`
- primitive array

LinkedList

list length (K)	1	7	63	511
performance (ns/op)	**7.25**	**9.03**	**20.87**	**29.07**
CPI (clockticks/instrn)	0.32	0.41	0.93	1.33
	events/operation			
cycles	17.97	22.77	51.32	72.66
instructions	56.08	55.88	54.96	54.49
L1-dcache-load-misses	1.18	1.83	1.87	2.65
L1-dcache-loads	18.94	19.39	18.88	18.22
L1-dcache-stores	12.00	12.18	11.99	11.15
LLC-load-misses	0	0	0.41	1.31
LLC-loads	0	0.72	1.33	1.56
dTLB-load-misses	0	0	0	0.90
dTLB-loads	19.05	19.00	19.15	18.09
dTLB-stores	12.04	12.09	12.16	11.02

What’s Going On?

`LinkedList`: node size is 24 bytes

Running on Intel Core i5:

L1 data	32K
L2	256K
L3	3M

Each new list item is 40 bytes (24 + 16)

- L1 cache will be full at <1K items

`ArrayList` is better, but not much: each new item is 20 bytes

Primitive Array

list length (K)	1	7	63	511
performance (ns/op)	3.62	3.65	3.65	3.66
CPI	0.30	0.30	0.30	0.31
	events/operation			
cycles	9.09	9.16	9.10	9.13
instructions	30.24	30.13	29.94	29.85
L1-dcache-load-misses	0.00	0.01	0.06	0.06
L1-dcache-loads	12.00	12.00	11.97	12.14
L1-dcache-stores	6.00	6.02	6.02	6.04
LLC-load-misses	0.00	0.00	0.00	0.00
LLC-loads	0.00	0.00	0.00	0.00
dTLB-load-misses	0.00	0.00	0.00	0.00
dTLB-loads	12.17	12.00	11.90	12.06
dTLB-stores	6.03	5.99	5.98	6.05

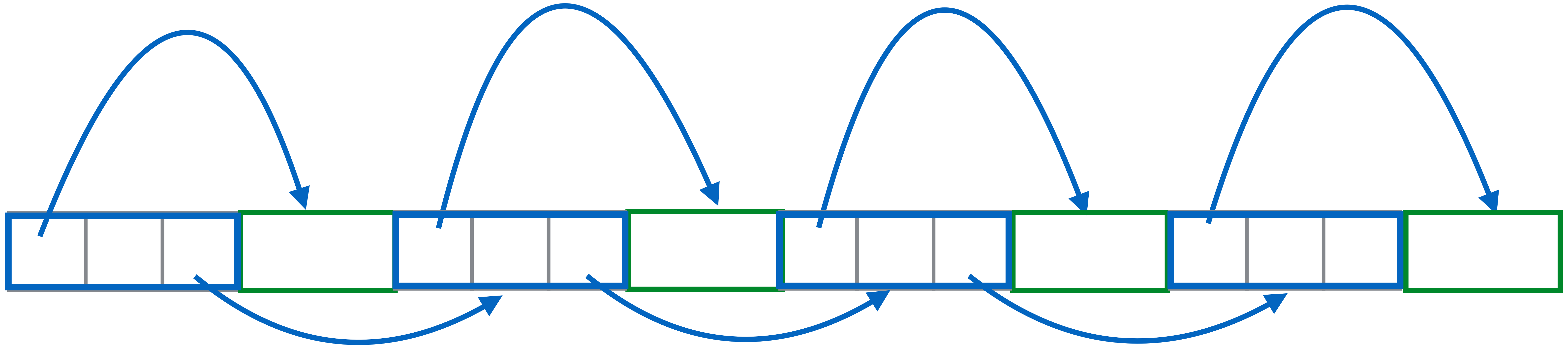
How Does Caching Play With Complexity?

Second issue: data locality

Two different problems:

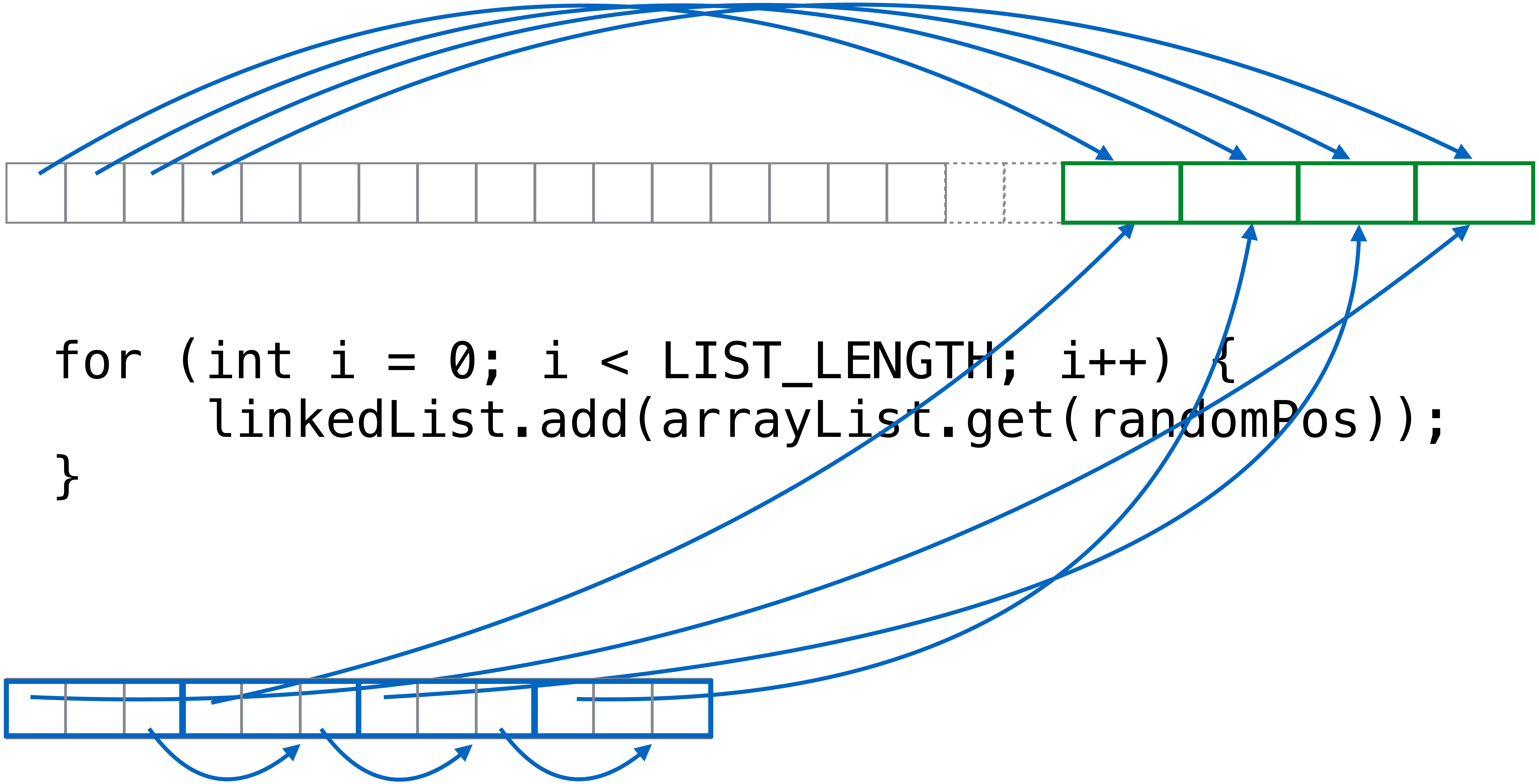
- Data density: how much of your 64-byte cache lines is data that you actually need?
- Prefetching: are you giving the processor a chance to help?

Populating LinkedList "Naturally"



```
for (int i = 0; i < LIST_LENGTH; i++) {
      linkedList.add(random.nextInt());
}
```

Populating LinkedList Randomly

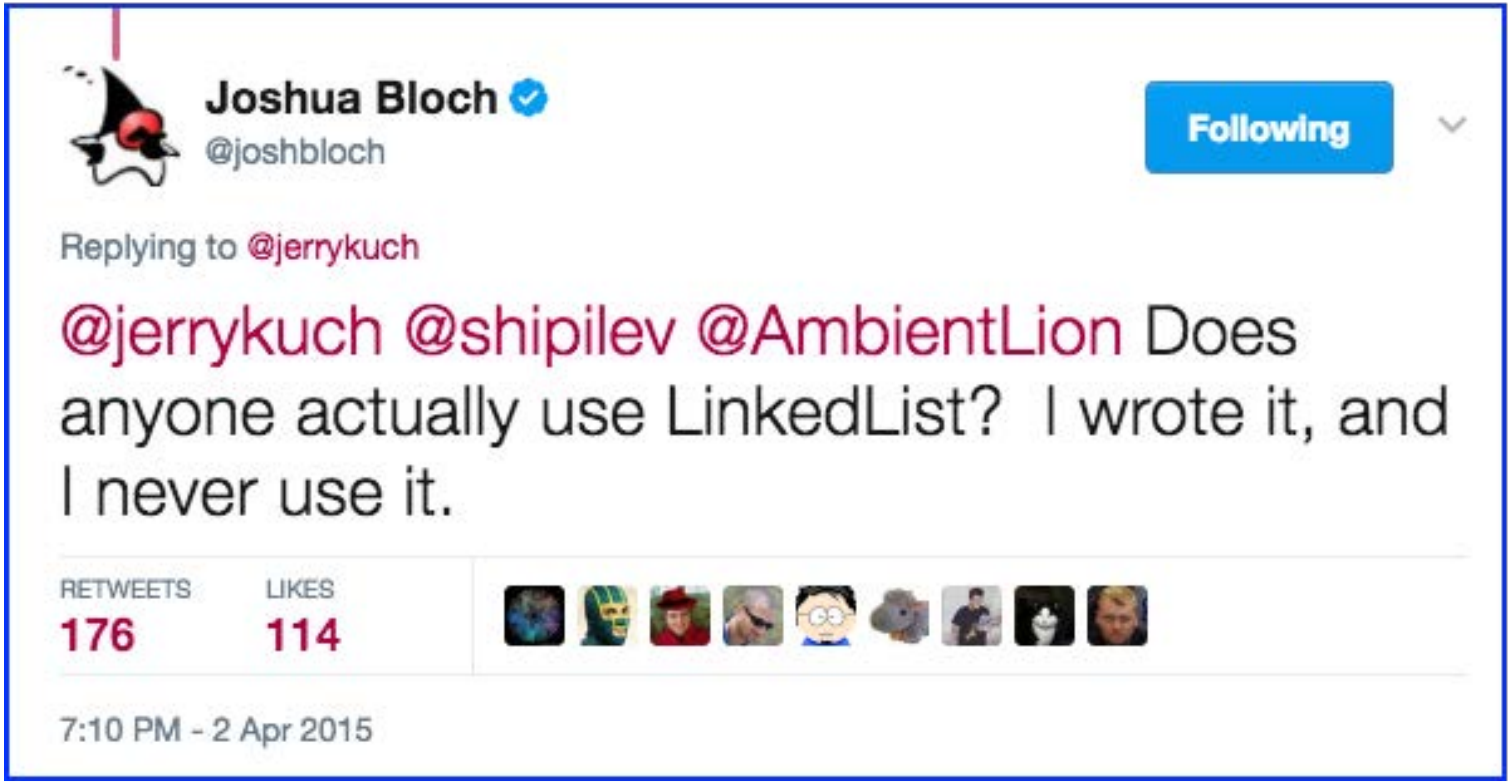


```
for (int i = 0; i < LIST_LENGTH; i++) {
     linkedList.add(arrayList.get(randomPos));
}
```

“Demo”

```
hosea-2:solutions mpn$ 
```

Poor Unloved `LinkedList`...



Joshua Bloch
@joshbloch

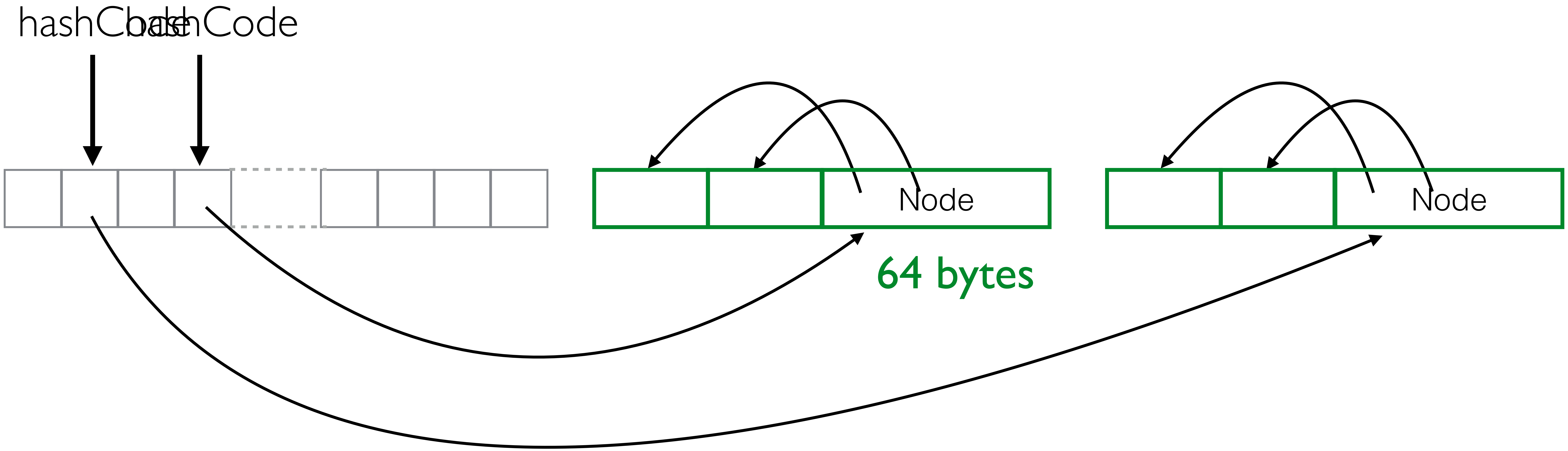
Replying to @jerrykuch

@jerrykuch @shipilev @AmbientLion Does anyone actually use LinkedList? I wrote it, and I never use it.

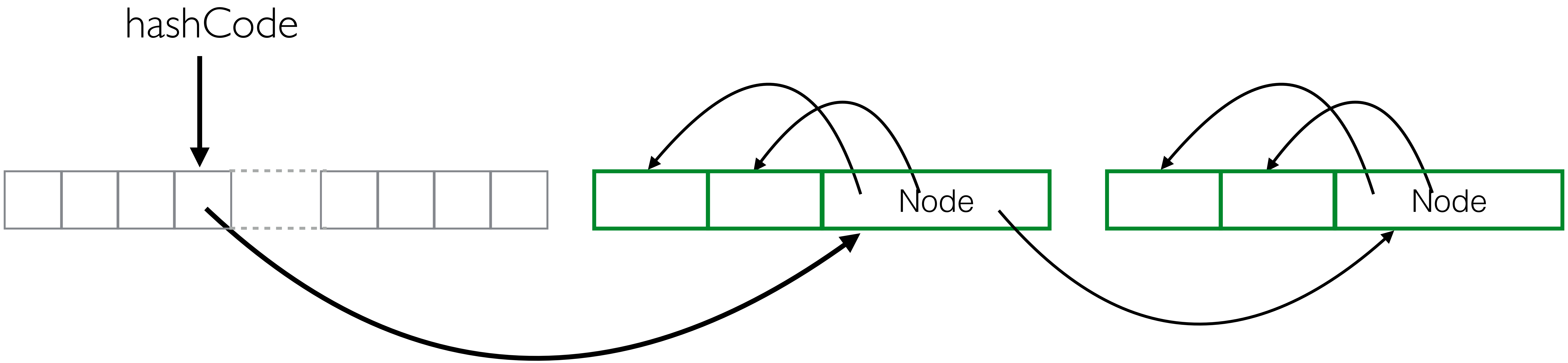
RETWEETS 176 LIKES 114

7:10 PM - 2 Apr 2015

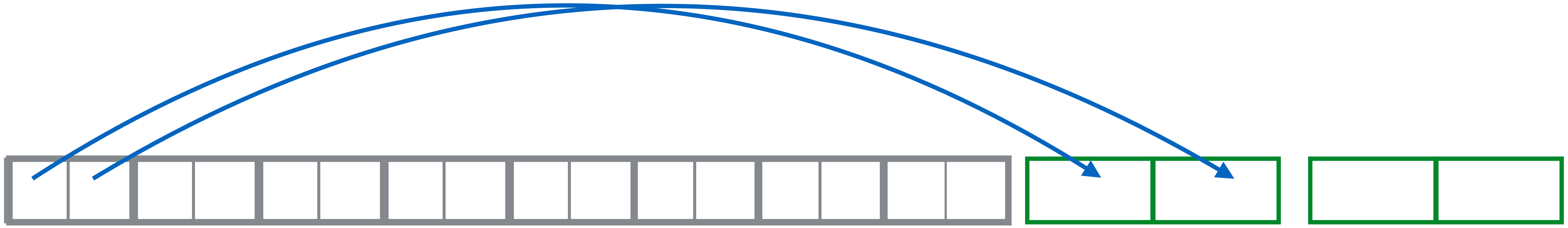
HashMap



HashMap collision



ImmutableCollections.MapN



```
Map<Integer,Integer> immutableMap = Map.ofEntries(
    Map.entry(512,1024),
    Map.entry(513,1026),
    Map.entry(514,1028));
```

Reducing Memory Footprint

Third-party collections frameworks usually have a focus on low memory footprint:

- Eclipse Collections
- fastutil
- Vavr (formerly Javaslang)
- Apache Commons Collections
- Guava
- Trove
- Argon

Improving Data Locality

- 3rd-party frameworks (previous slide) often support primitive collections
- ObjectLayout
  - `StructuredArray` – like a C-style "array of struct"
  - also arrays as part of objects, and cohered aggregates
- Roaring Bitmaps
  - compressed bitmaps, very fast
- Project Valhalla
  - language-level solution for value objects and primitive collections

Conclusion, of sorts…

Performance **mostly doesn’t matter**

… but when it does matter, it really matters!

Every performance improvement represents a tradeoff

Algorithm complexity is still important

… but so is

- network/database performance,
- GC,
- resource contention,
- caching

So actually it **is** your father’s complexity – just a lot more complex than before!