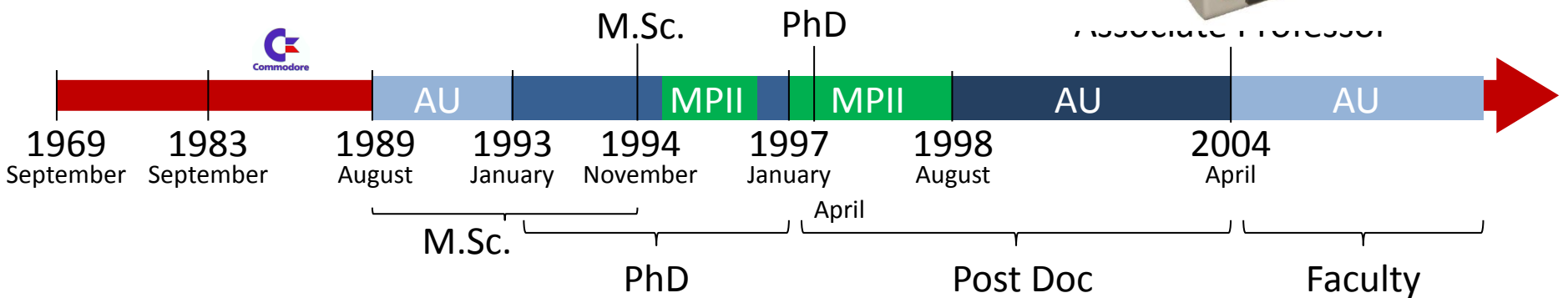


Massive Data Algorithmics


Gerth Stølting Brodal
Aarhus University



Gerth Stølting Brodal



Outline of Talk

- 
 - who, where, what ?
 - reseach areas
- External memory algorithmics
 - models
 - searching and sorting
- Fault-tolerant searching
- Flow simulation

maDaLGO 
CENTER FOR MASSIVE DATA ALGORITHMICS

- Where?



madaligo 
CENTER FOR MASSIVE DATA ALGORITHMICS





Danmarks
Grundforskningsfond
Danish National
Research Foundation

- Center of
- **Lars Arge**, Professor, Centerleader
- Gerth S. Brodal, Associate Professor
- 5 Post Docs, 10 PhD students, 4 TAP
- Total budget for 5 years ca. 60 million DKR

AU



Arge



Brodal

MIT



Demaine



Indyk

MPII



Mehlhorn

Frankfurt



Meyer

Faculty

Lars Arge

Gerth Stølting Brodal

Researchers



Henrik Blunck



Brody Sandel



Nodari Sitchinava



Elad Verbin



Qin Zhang

PhD Students

Lasse Kosetski Deleuran



Freek van Walderveen

Casper Kejlberg-Rasmussen

Kasper Dalgaard Larsen

Jesper Erenskjold Moeslund

Jakob Truelsen



Kostas Tsakalidis

Mark Greve

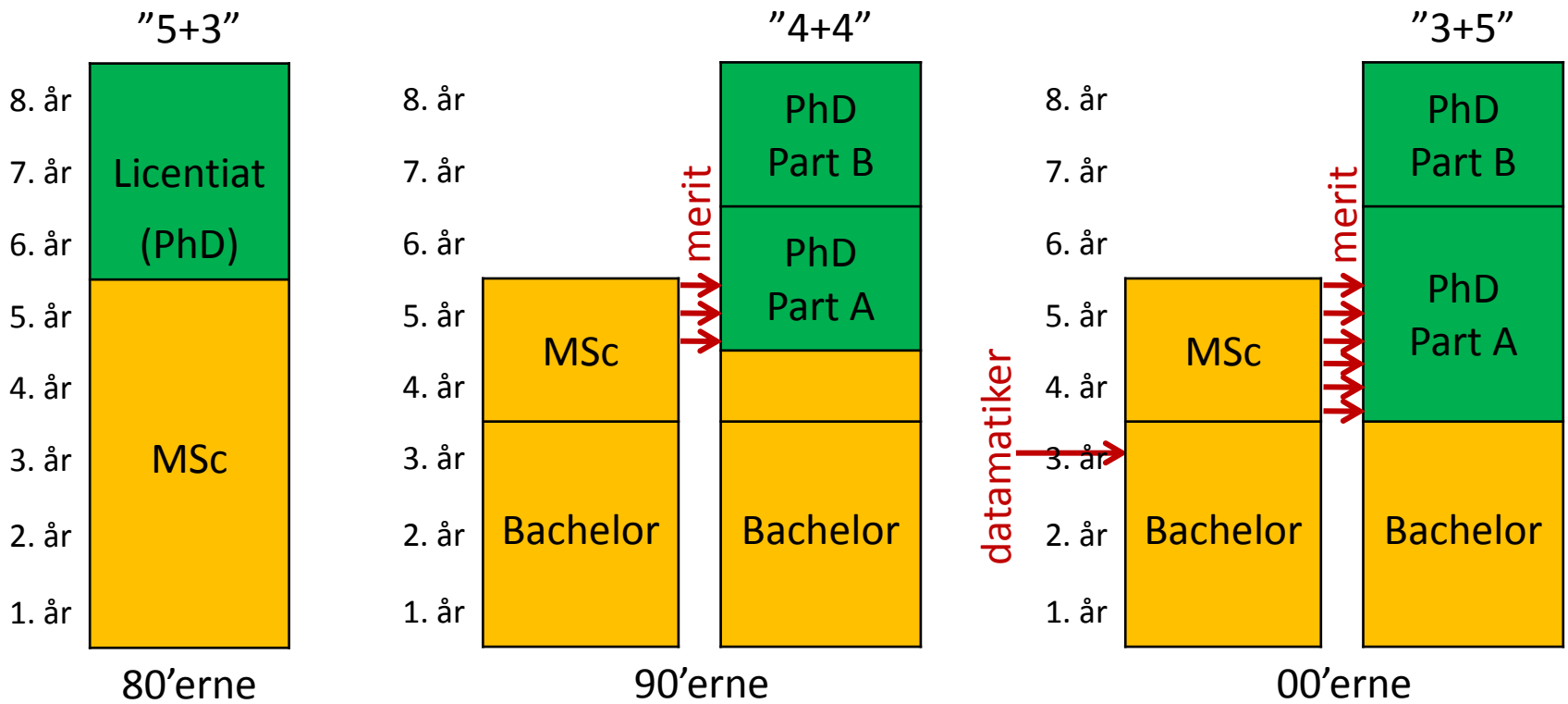
Morten Revsbæk



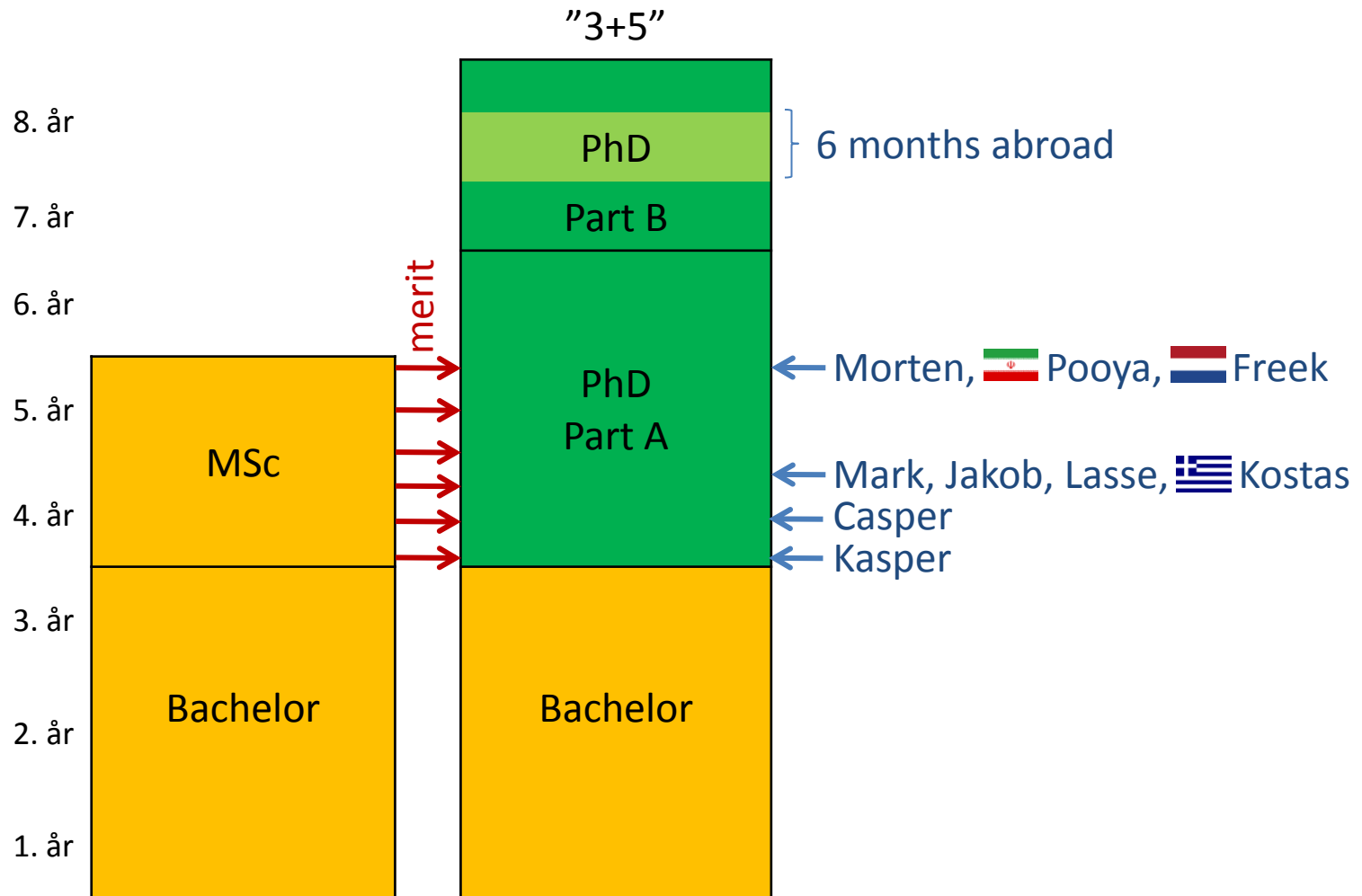
Pooya Davoodi



PhD Education @ AU



PhD Education @ MADALGO





MASSIVE 2010



Summer School 2010



SoCG 2009



Retreat 2009



Retreat 2008



MASSIVE 2009



Summer School 2008



Retreat 2007



Summer School 2007

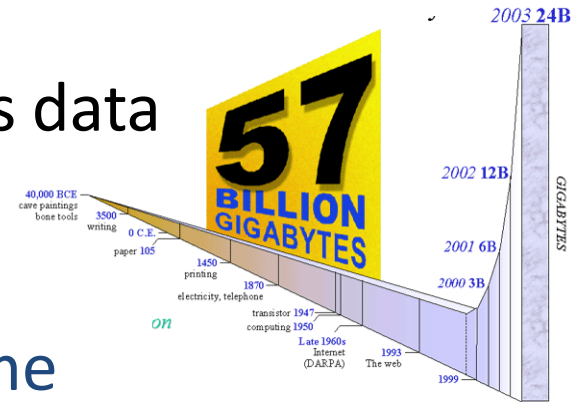


Inauguration 2007

- *High level objectives*
 - Advance algorithmic knowledge in “massive data” processing area
 - Train researchers in world-leading international environment
 - Be catalyst for multidisciplinary/industry collaboration
- *Building on*
 - Strong international team
 - Vibrant international environment (focus on people)

Massive Data

- Pervasive use of computers and sensors
- Increased ability to acquire/store/process data
→ Massive data collected everywhere
- Society increasingly “data driven”
→ Access/process data anywhere any time



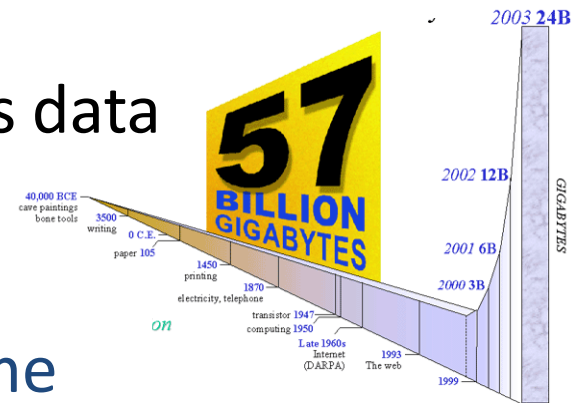
Nature special issues

- 2/06: “2020 – Future of computing”
- 9/08: “BIG DATA”
- Scientific data size growing exponentially, while quality and availability improving
- Paradigm shift: *Science will be about mining data*
→ Computer science paramount in all sciences



Massive Data

- Pervasive use of computers and sensors
- Increased ability to acquire/store/process data
→ Massive data collected everywhere
- Society increasingly “data driven”
→ Access/process data anywhere any time



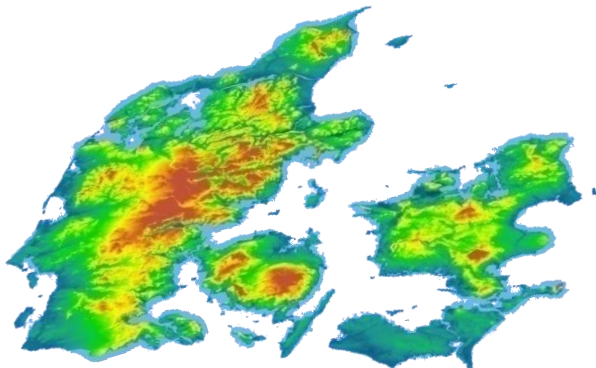
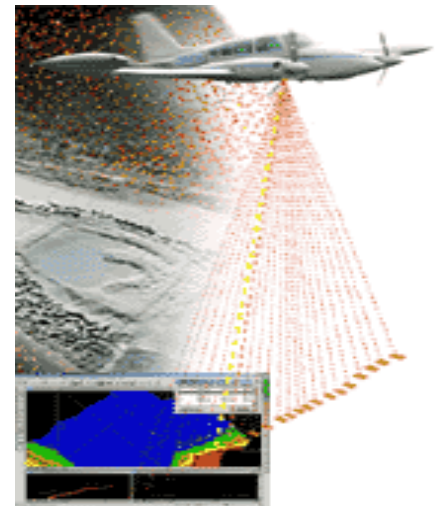
Obviously not only in sciences:

- Economist 02/10:
 - From 150 Billion Gigabytes five years ago to 1200 Billion today
 - Managing data deluge difficult; doing so will transform business/public life



Example: Massive Terrain Data

- New technologies: Much easier/cheaper to collect detailed data
 - **Previous** 'manual' or radar based methods
 - Often 30 meter between data points
 - Sometimes 10 meter data available
 - **New** laser scanning methods (**LIDAR**)
 - Less than 1 meter between data points
 - Centimeter accuracy (previous meter)

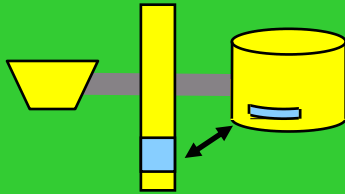


Denmark

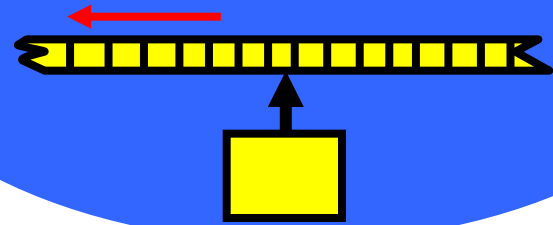
~2 million points at 30 meter (<1GB)

~18 billion points at 1 meter (>1TB)

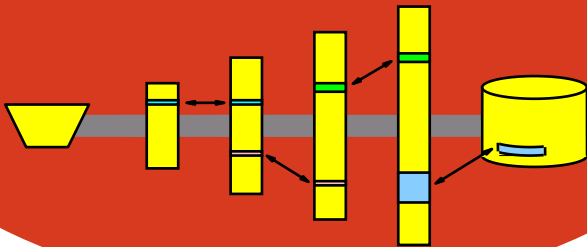
I/O Efficient Algorithms



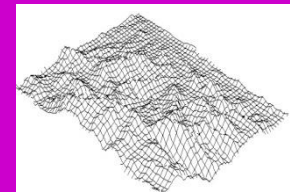
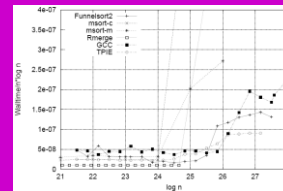
Streaming Algorithms



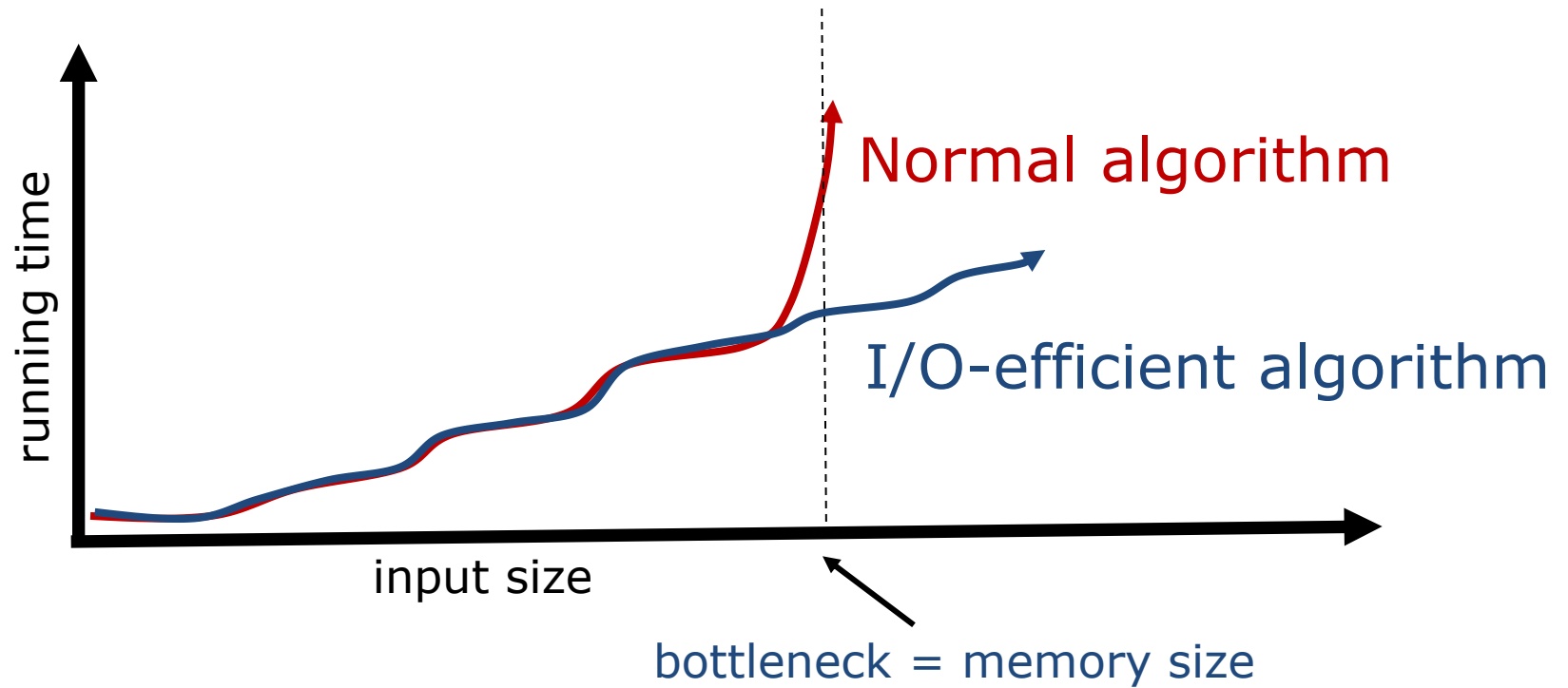
Cache Oblivious Algorithms



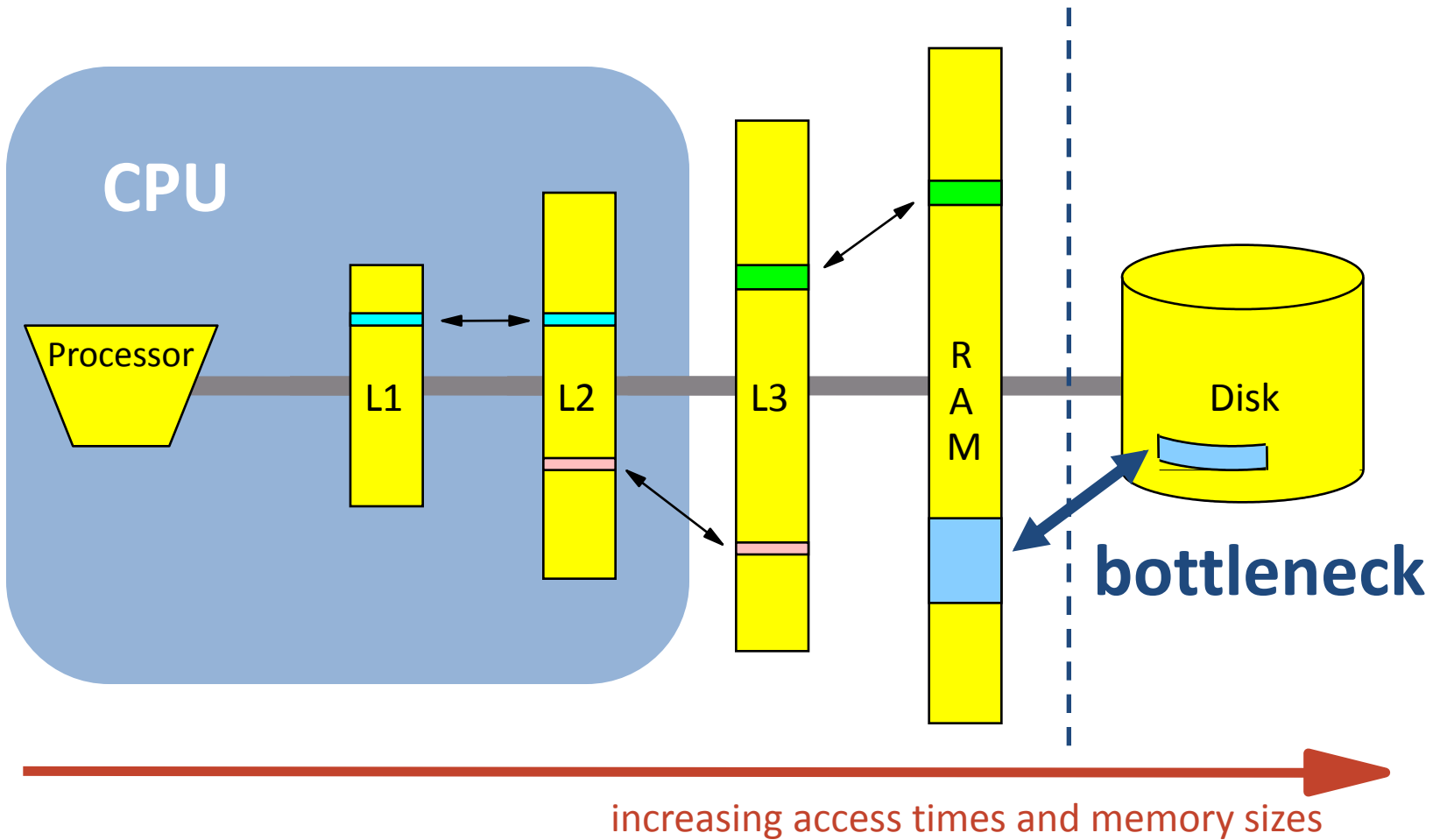
Algorithm Engineering



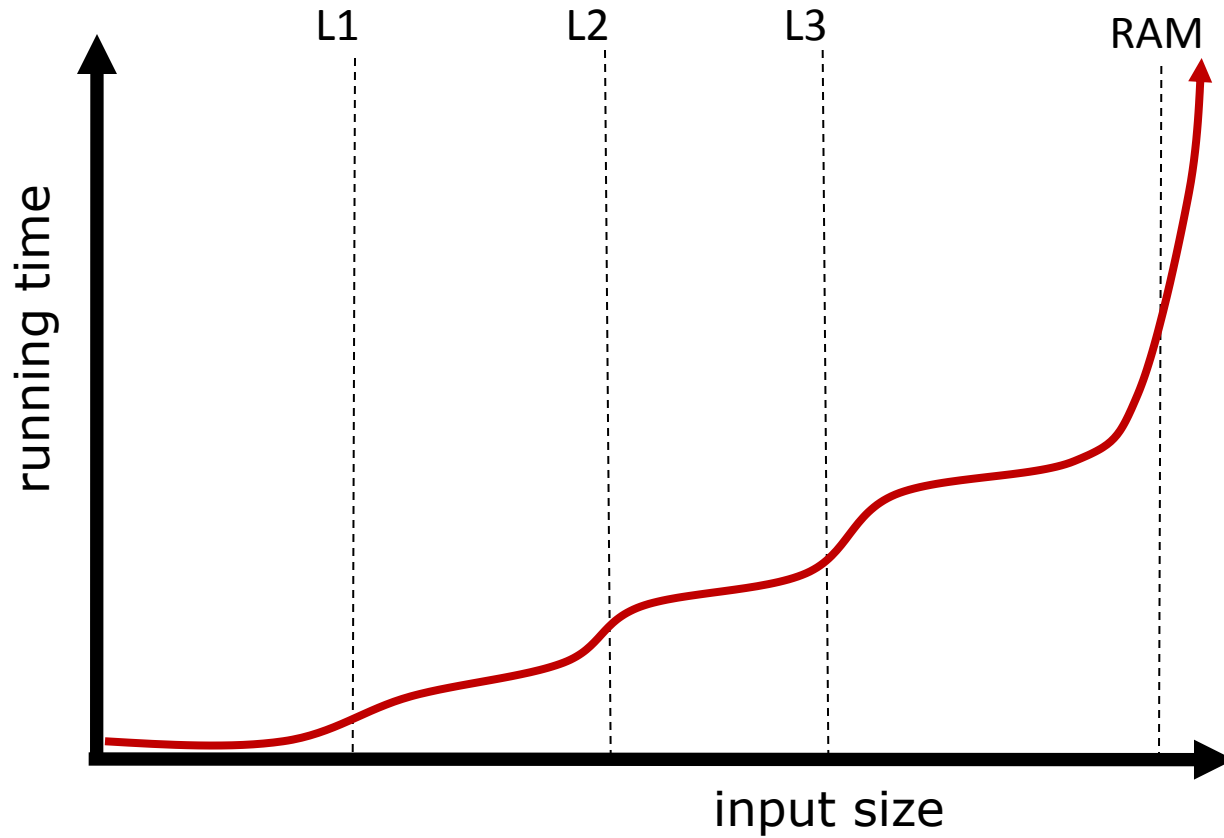
The problem...



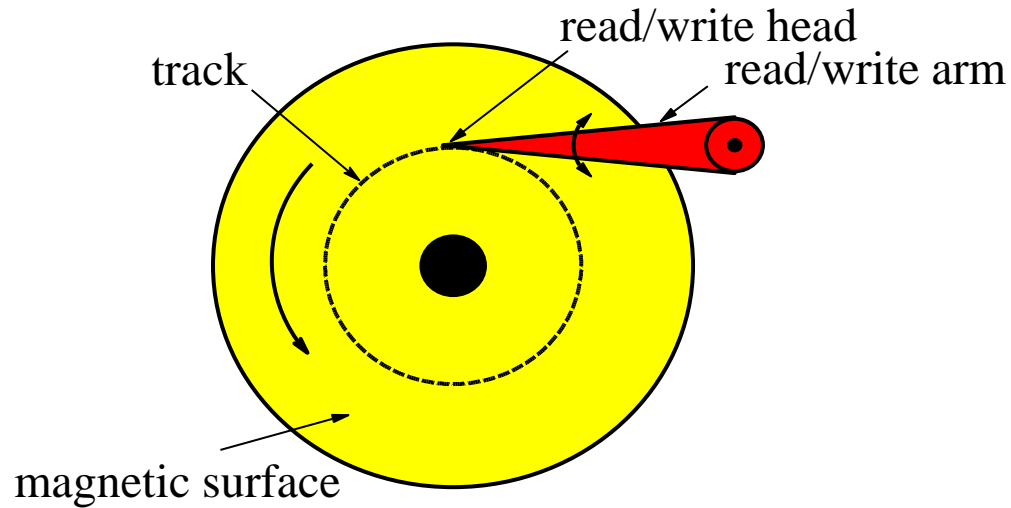
Memory Hierarchies



Memory Hierarkies vs. Running Time

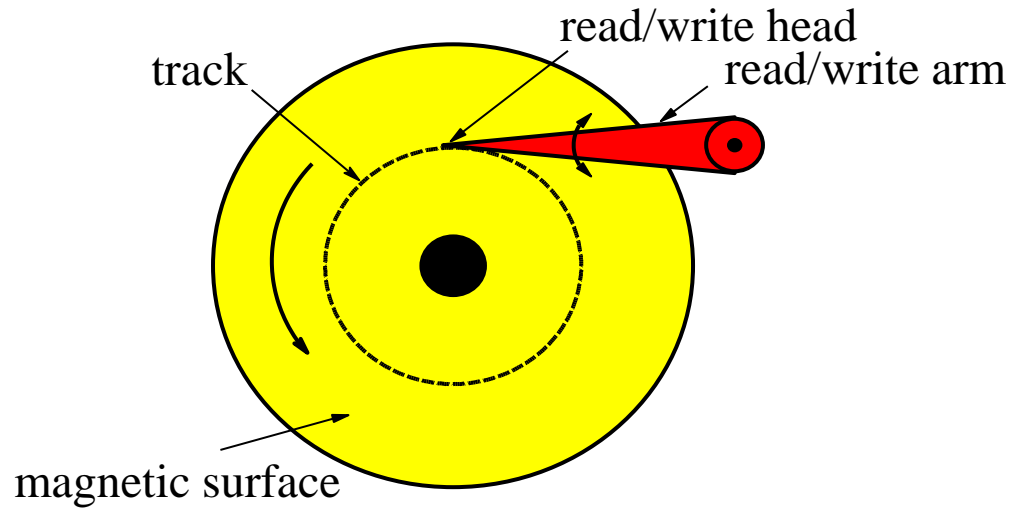


Disk Mechanics



"The difference in speed between modern CPU and disk technologies is analogous to the difference in speed in sharpening a pencil using a sharpener on one's desk or by taking an airplane to the other side of the world and using a sharpener on someone else's desk." (D. Comer)

Disk Mechanics



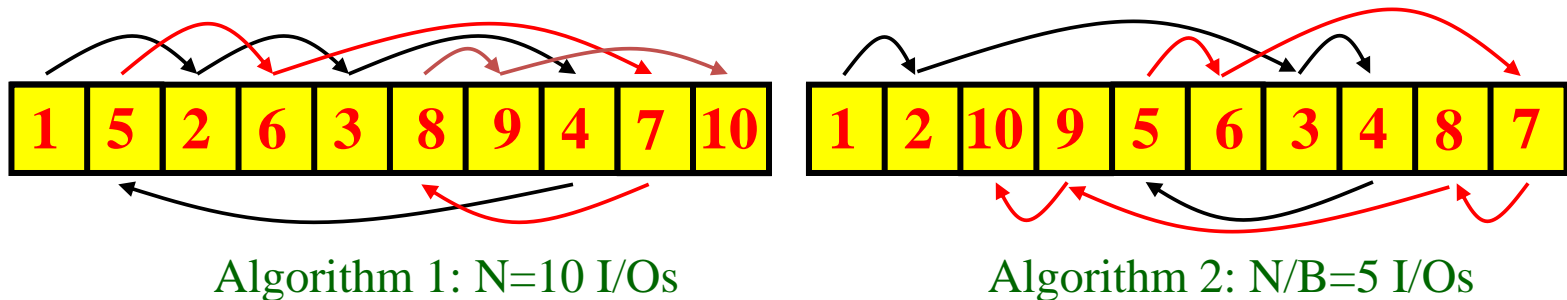
- I/O is often bottleneck when handling massive datasets
- Disk access is 10^7 times slower than main memory access!
- Disk systems try to amortize large access time transferring large contiguous blocks of data
- Need to store and access data to take advantage of blocks !

Memory Access Times

	Latency	Relative to CPU
Register	0.5 ns	1
L1 cache	0.5 ns	1-2
L2 cache	3 ns	2-7
DRAM	150 ns	80-200
TLB	500+ ns	200-2000
Disk	10 ms	10^7

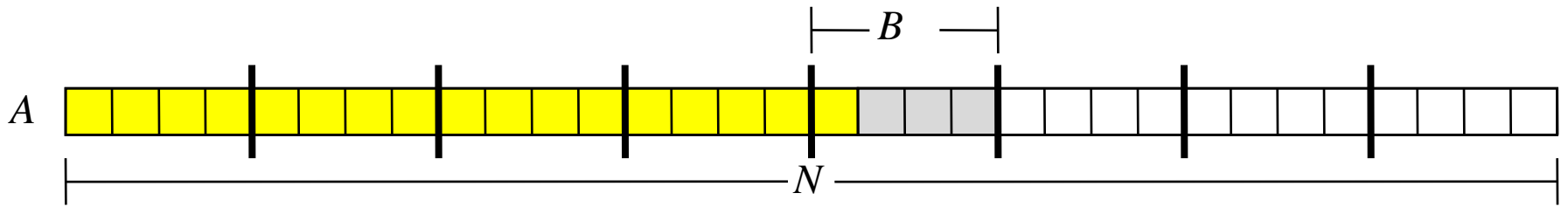
I/O-Efficient Algorithms Matter

- **Example:** Traversing linked list (List ranking)
 - Array size $N = 10$ elements
 - Disk block size $B = 2$ elements
 - Main memory size $M = 4$ elements (2 blocks)



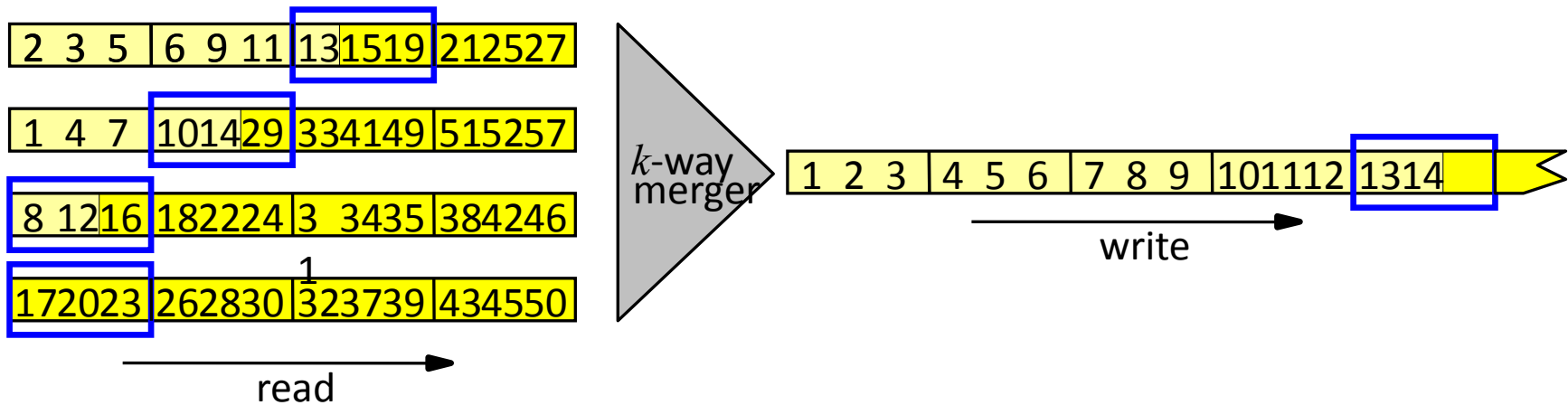
- Difference between N and N/B large since block size is large
 - **Example:** $N = 256 \times 10^6$, $B = 8000$, $1ms$ disk access time
 - $\Rightarrow N$ I/Os take 256×10^3 sec = 4266 min = **71 hr**
 - $\Rightarrow N/B$ I/Os take $256/8$ sec = **32 sec**

I/O Efficient Scanning



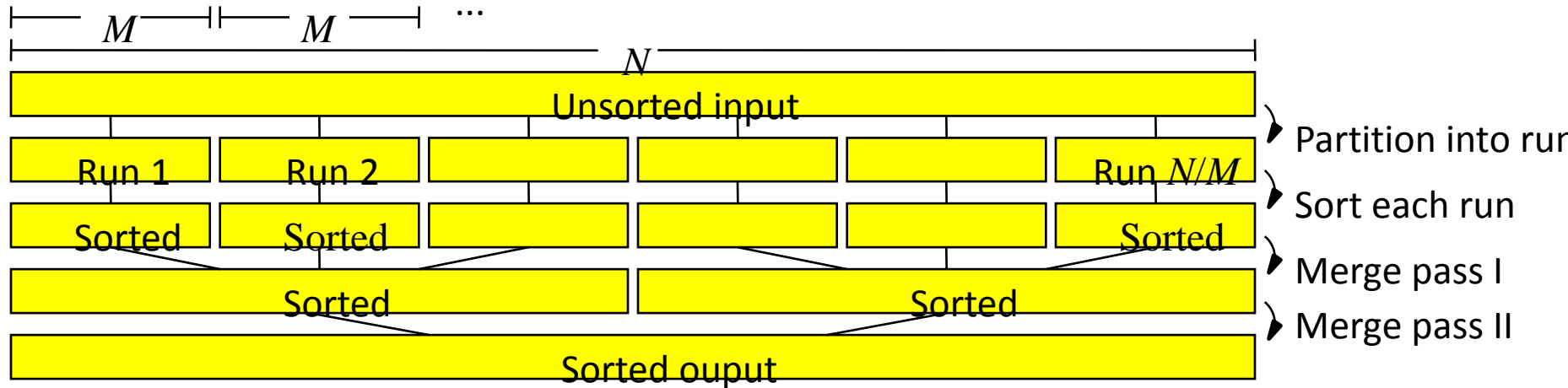
$O(N/B)$ I/Os

External-Memory Merging



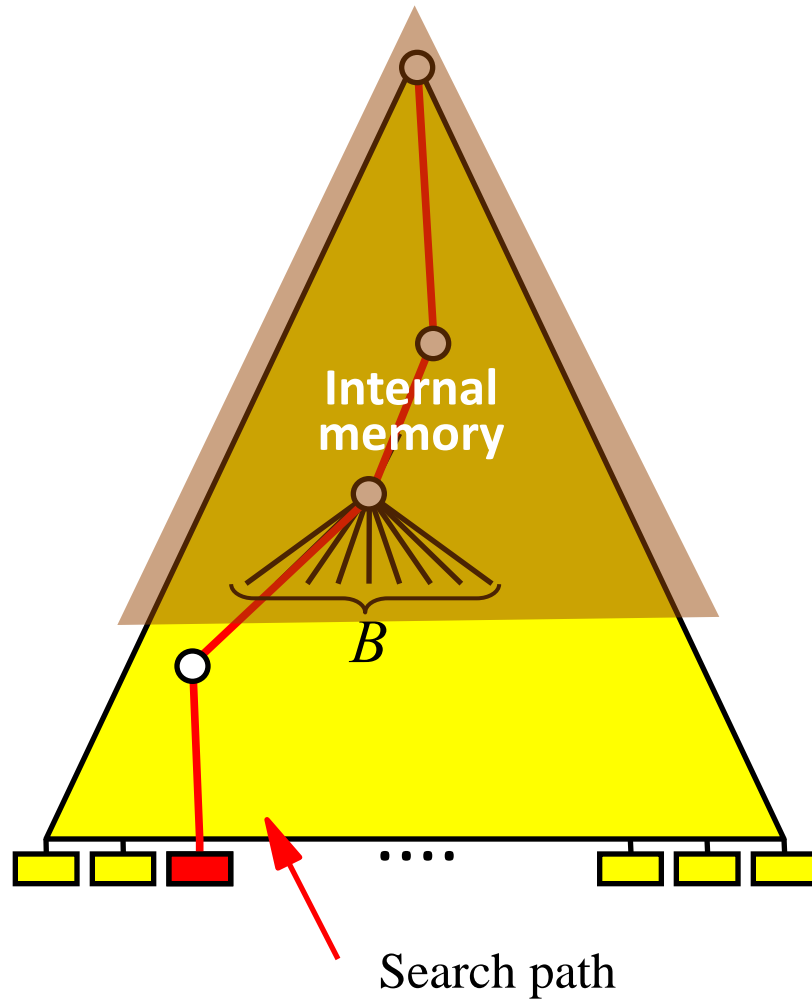
Merging k sequences with N elements requires $O(N/B)$ IOs
(provided $k \leq M/B - 1$)

External-Memory Sorting



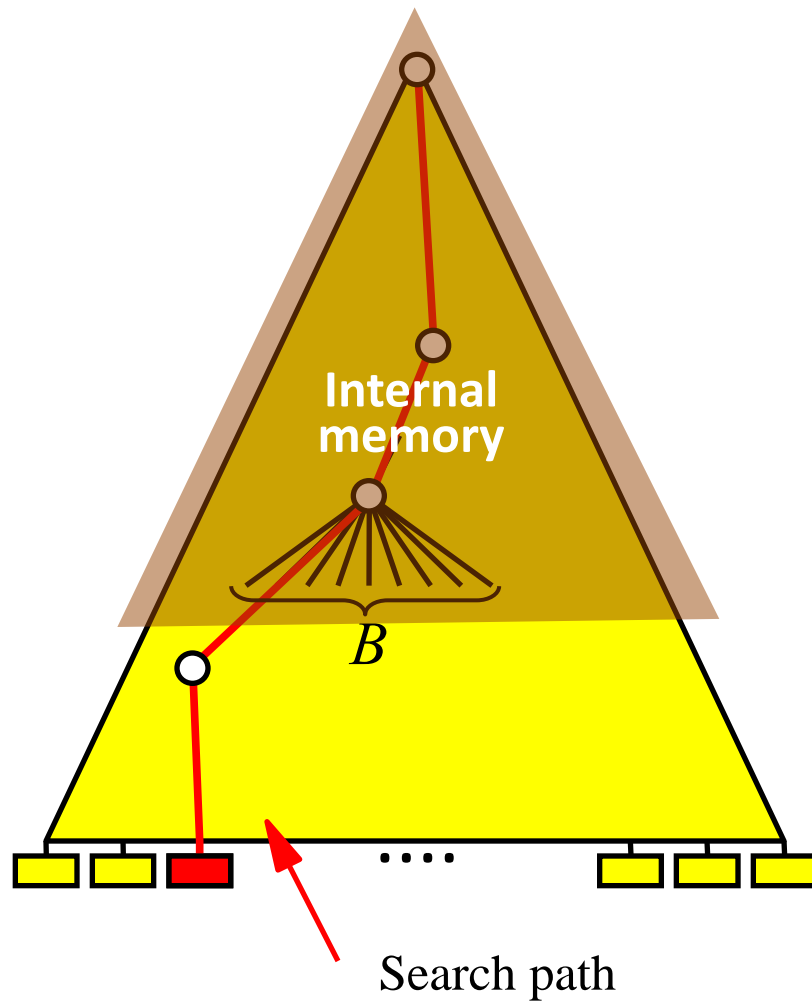
- MergeSort uses $O(N/B \cdot \log_{M/B}(N/B))$ I/Os
- Practice number I/Os: 4-6 x scanning input

B-trees - The Basic Searching Structure



- **Searches**
Practice: 4-5 I/Os
- **Repeated searching**
Practice: 1-2 I/Os

B-trees



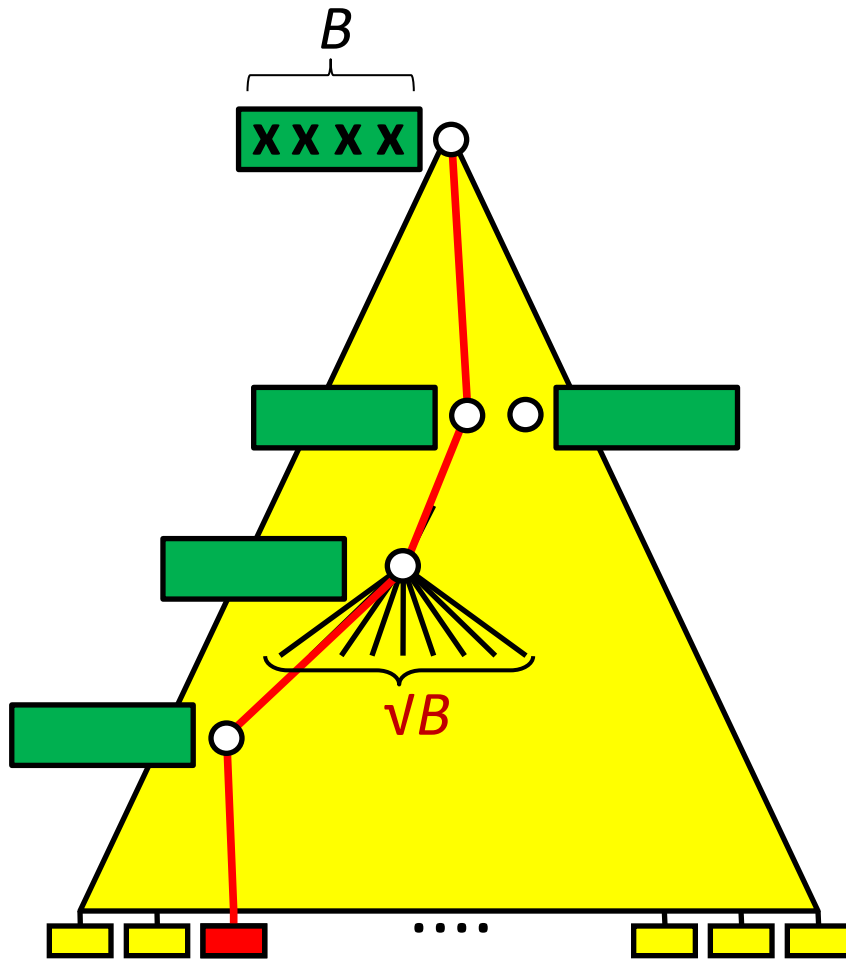
Best possible



- Searches $O(\log_B N)$ I/Os
- Updates $O(\log_B N)$ I/Os



B-trees with Buffered Updates



- Searches cost

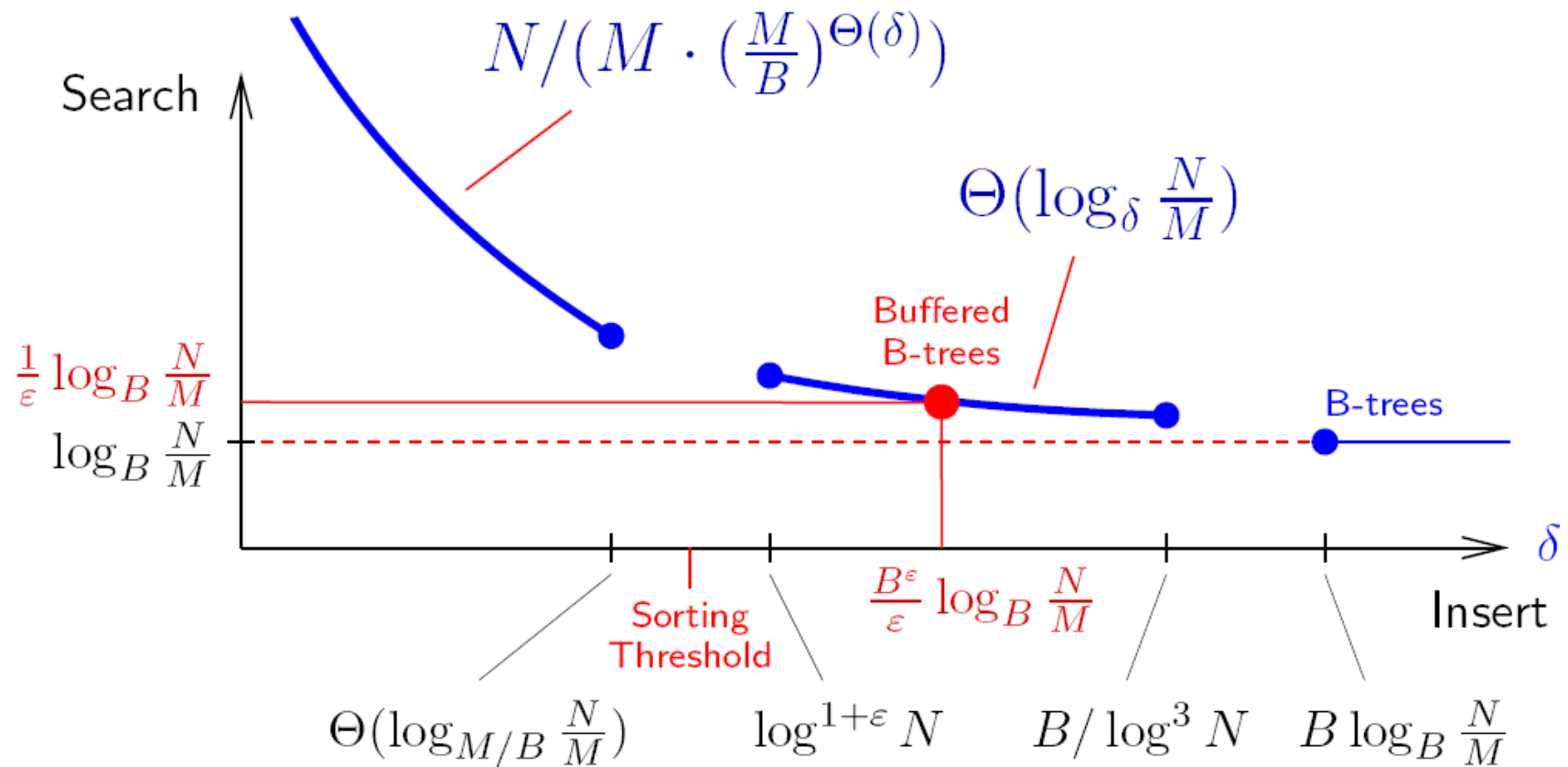
$$O(\log_B N) \text{ I/Os}$$

- N updates cost

$$O(N / \sqrt{B} \cdot \log_B N) \text{ I/Os}$$

Trade-off between search and update times – optimal !

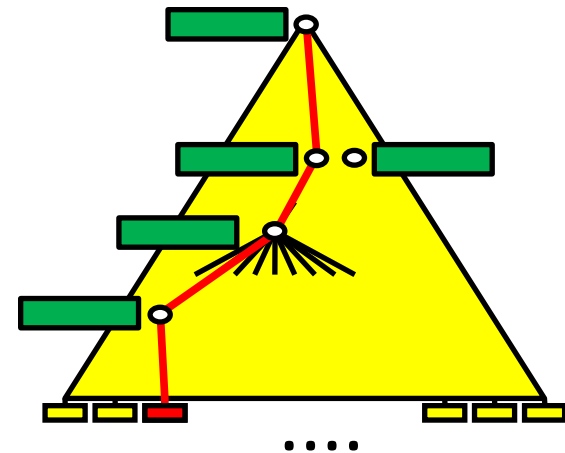
B-trees with Buffered Updates



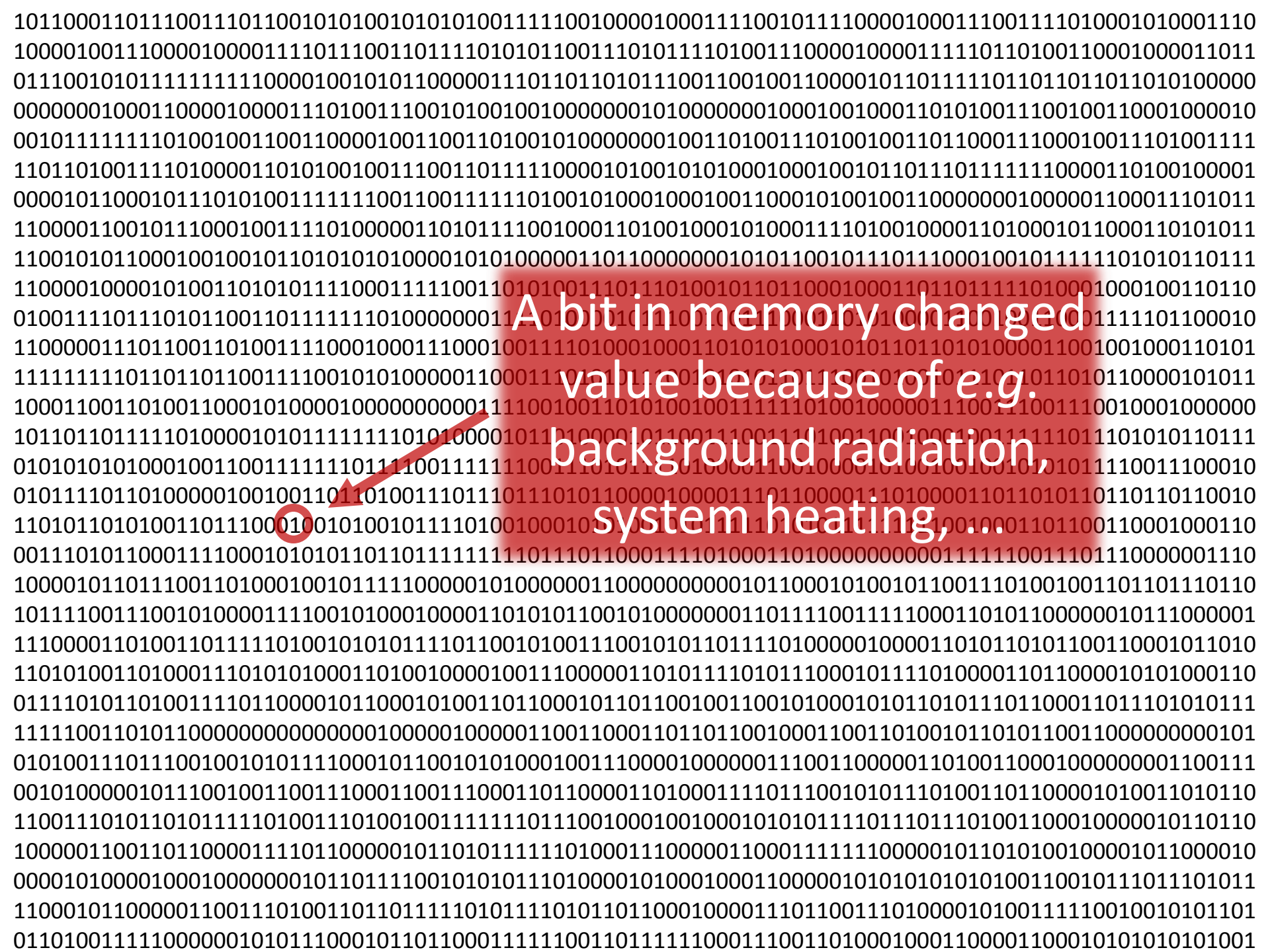
B-trees with Buffered Updates

Experimental Study

- 100.000.000 elements
- Search time basically unchanged with buffers
- Updates 100 times faster



101100011011100111011001010100101010100111110010000100011110010111100001000111001111010001010001110
100001001110000100001111011100110111101010110011101011110100111000010000111110110100110001000011011
011100101011111111110000100101011000001110110110101110011001001100001011011111011011011011010100000
000000010001100001000011101001110010100100100000001010000000100010010001101010011100100110001000010
00101111111010010011001100001001100110100101000000010011010011101001001101100011100010011101001111
11011010011110100001101010010011100110111110000101001010100010001001011011101111110000110100100001
00001011000101110101001111111001100111111010010100010001001100010100100110000001000001100011101011
110000110010111000100111101000001101011110010001101001000101000111101001000011010001011000110101011
110010101100010010010110101010100001010100000110110000000101011001011101110001001011111101010110111
110000100001010011010101111000111110011010100111011101001011011000100011011011111010001000100110110
010011110111010110011011111110100000001111010000101110010011100011010100001100100110001111101100010
110000011101100110100111100010001110001001111010001000110101010001010110110101000011001001000110101
111111111011011011001111001010100000110001110101011100101010110111001010010111011011010110000101011
10001100110100110001010000100000000001111001001101010010011111101001000001110011100111001000100000
101101101111101000010101111111101010000101101000010110011100111010011001000100111111011101010110111
0101010101000100110011111111011110011111110011101011110100001100100001010010010010101011110011100010
010111101101000001001001101101001110111011101011000010000111011000011101000011011010110110110110010
110101101010011011100000010100101111010010001010100101011111010101111111110010001101100110001000110
00111010110001111000101010110110111111110111011000111101000110100000000001111110011101110000001110
100001011011100110100010010111110000010100000011000000000010110001010010110011101001001101101110110
10111100111001010000111100101000100001101010110010100000011011110011111000110101100000010111000001
111000011010011011111010010101011110110010100111001010110111101000001000011010110101100110001011010
110101001101000111010101000110100100001001110000011010111101011100010111101000011011000010101000110
011110101101001111011000010110001010011011000101101100100110010100010101101011101100011011101010111
11111001101011000000000000000000000000100000100000110011000110110110010001100110100101101011001100000000101
010100111011100100101011110001011001010100010011100001000000111001100000110100110001000000001100111
001010000010111001001100111000110011100011011000011010001111011100101011101001101100001010011010110
110011101011010111110100111010010011111110111001000100100010101011110111011101001100010000010110110
100000110011011000011110110000010110101111110100011100000110001111111000001011010100100001011000010
000010100001000100000001011011110010101011101000010100010001100000101010101010100110010111011101011
110001011000001100111010011011011111010111101011011000100001110110011101000010100111110010010101101
01101001111100000010101110001011011000111111000111001101000100011000011000011000101010101001



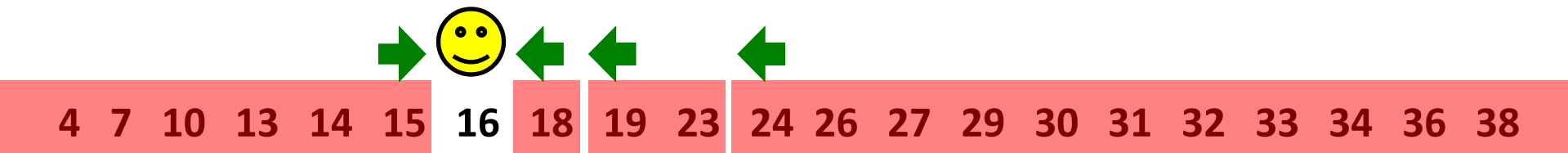
A bit in memory changed
value because of e.g.
background radiation,
system heating, ...



"You have to provide reliability on a software level. If you're running 10,000 machines, something is going to die every day."

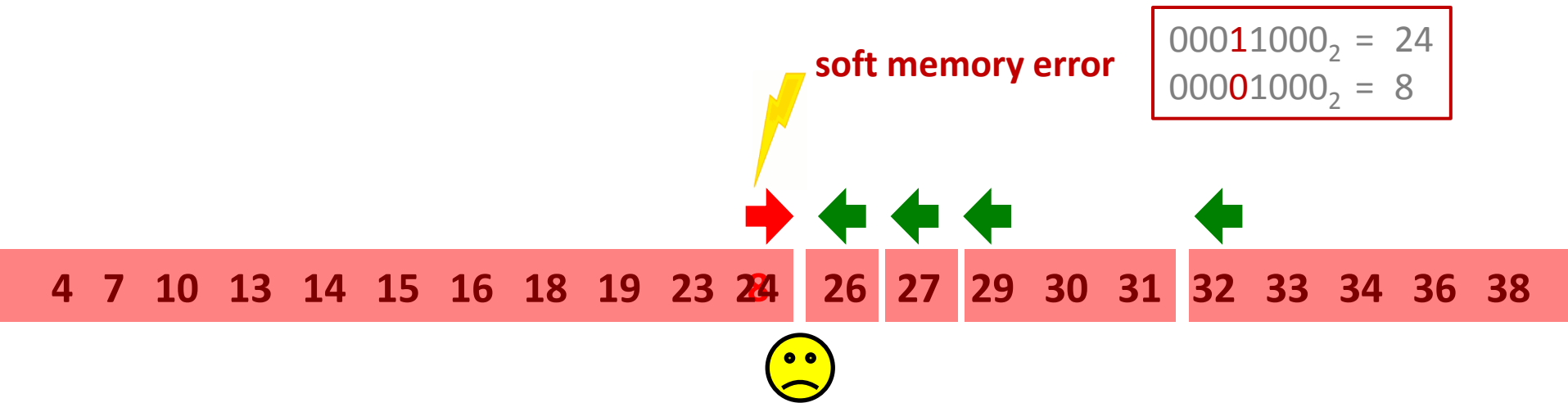
—  fellow Jeff Dean

Binary Search for 16



$O(\log N)$ comparisons

Binary Search for 16



Requirement: If the search key occurs in the array as an uncorrupted value, then we should report a match !

Where is Michael ?



Where is Michael ?



Where is Michael ?

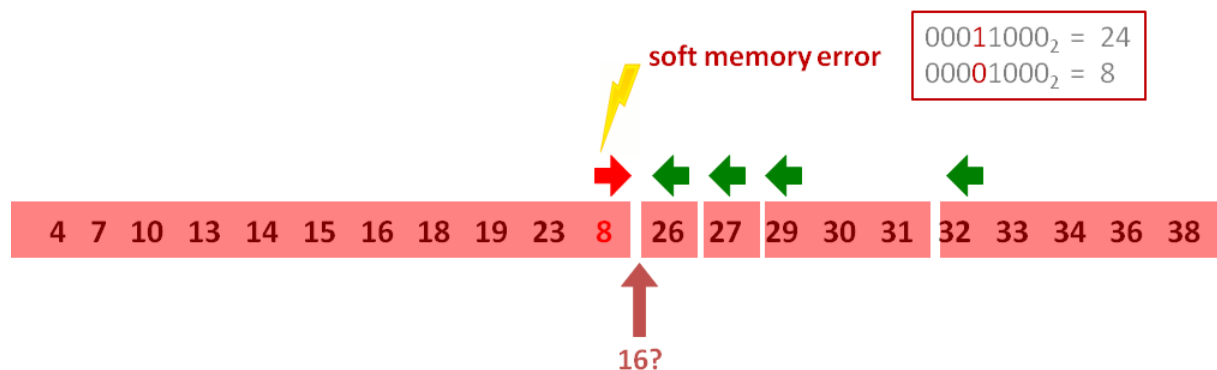


If at most 4 faulty answers then Jens is somewhere here

Faulty-Memory RAM Model

Finocchi and Italiano, STOC'04

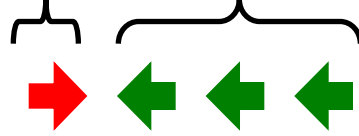
- **Content** of memory cells can get **corrupted**
- Corrupted and uncorrupted content **cannot be distinguished**
- $O(1)$ **safe** registers
- **Assumption**: At most δ corruptions



Faulty-Memory RAM: Searching

Problem?

Low confidence High confidence



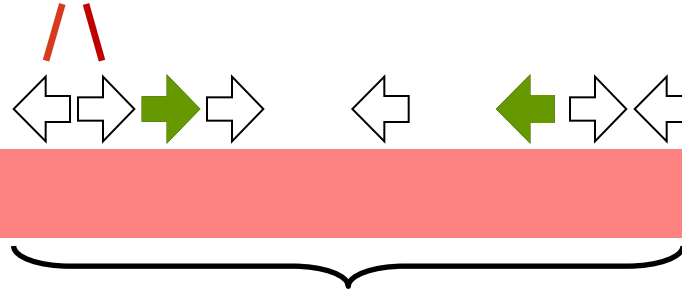
4 7 10 13 14 15 16 18 19 23 8 26 27 29 30 31 32 33 34 36 38

↑
16?

Faulty-Memory RAM: Searching

When are we
done ($\delta=3$)?

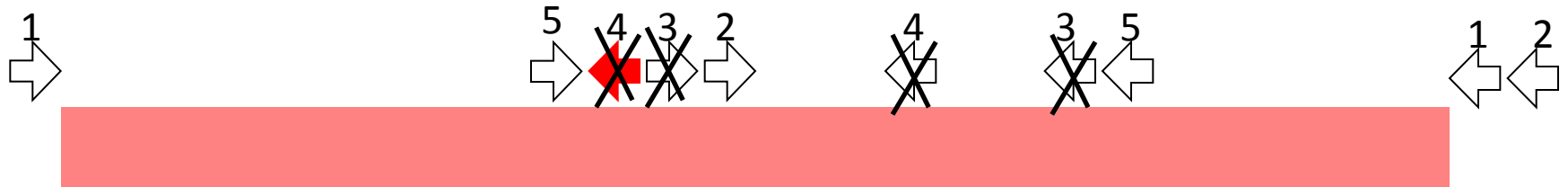
Contradiction, i.e. at
least one fault



If range contains at least $\delta+1 \rightarrow$ and $\delta+1 \leftarrow$ then there is
at least one uncorrupted \rightarrow and \leftarrow , i.e. x must be
contained in the range

Faulty-Memory RAM: $\Theta(\log N + \delta)$ Searching

Brodal, Fagerberg, Finocchi, Grandoni, Italiano, Jørgensen, Moruz, Mølhave, ESA'07



If verification fails

- contradiction, i.e. ≥ 1 memory-fault
- ignore 4 last comparisons
- **backtrack** one level of search



Flood Prediction Important

- Prediction areas susceptible to floods
 - Due to e.g raising sea level or heavy rainfall
- **Example:** Hurricane Floyd Sep. 15, 1999



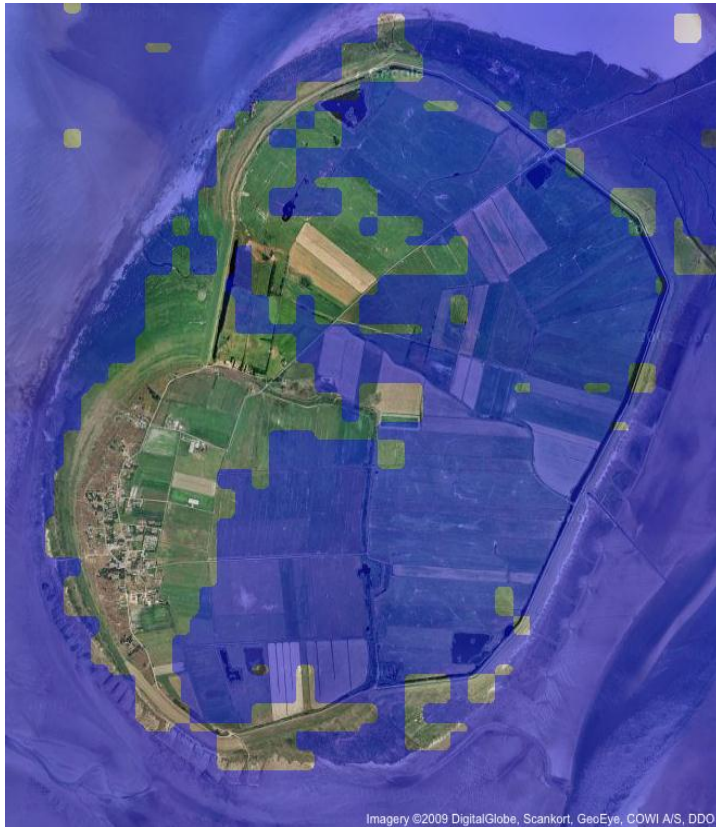
7 am



3pm

Detailed Terrain Data Essential

Mandø with 2 meter sea-level raise

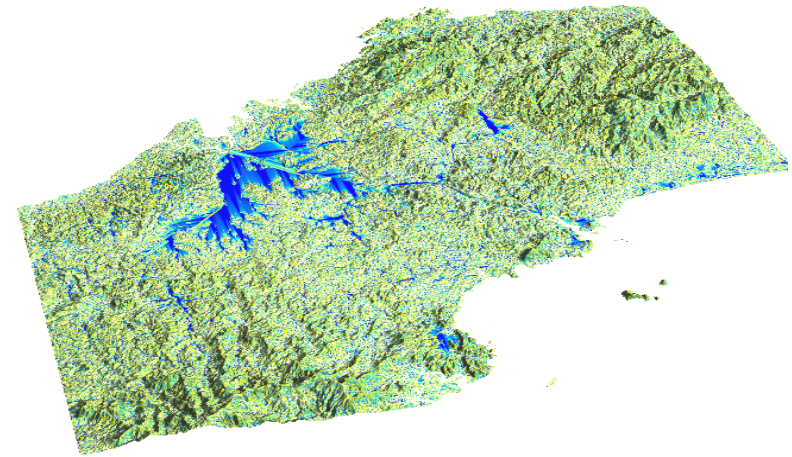


80 meter terrain model



2 meter terrain model

Surface Flow Modeling

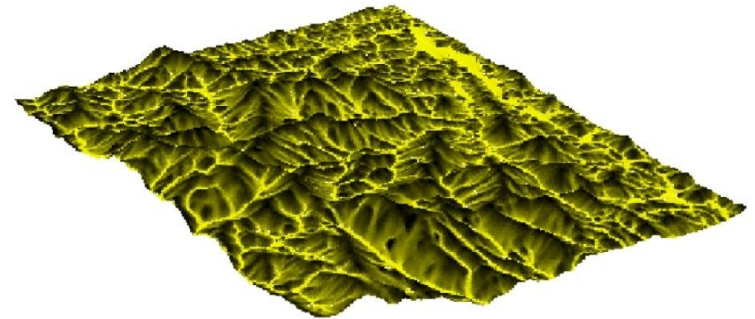
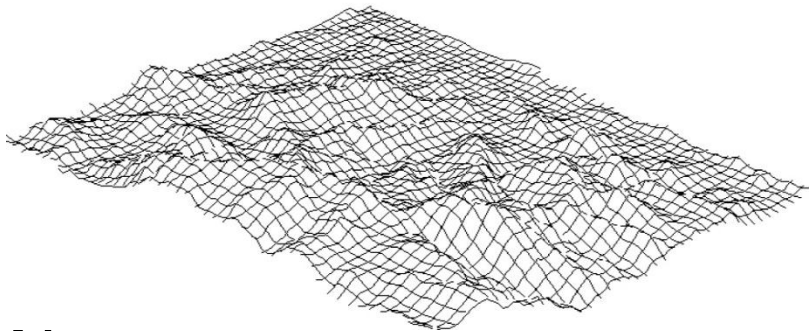


Hurricane Floyd (September 15, 1999)

- Conceptually flow is modeled using two basic attributes
 - **Flow direction**: The direction water flows at a point
 - **Flow accumulation**: Amount of water flowing through a point

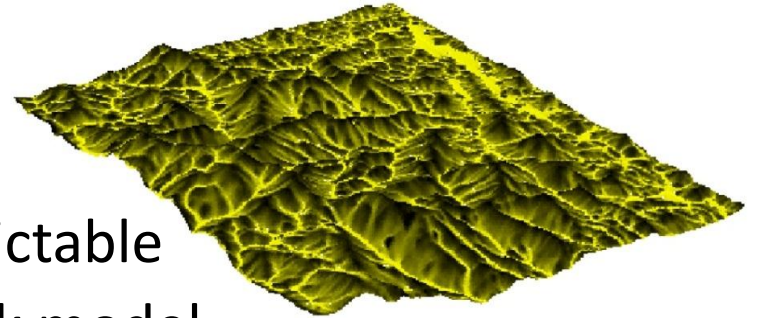
Flow Accumulation

- **Flow accumulation** on grid terrain model:
 - Initially one unit of water in each grid cell
 - Water (initial and received) distributed from each cell to lowest lower neighbor cell (if existing)
 - Flow accumulation of cell is total flow through it



- **Note:**
 - Flow accumulation of cell = size of “upstream area”
 - Drainage network = cells with high flow accumulation

Massive Data Problems



- Commercial systems:
 - Often *very* slow
 - Performance somewhat unpredictable
 - Cannot handle 2-meter Denmark model
- Collaboration environmental researchers in late 90'ties
 - US Appalachian mountains dataset
 - 800x800km at 100m resolution \Rightarrow a few Gigabytes **14 days!!**
 - Customized software on $\frac{1}{2}$ GB machine:
- Appalachian dataset would be Terabytes sized at 1m resolution!

Flood Modeling

- Not all terrain below height h is flooded when water rise to h meters!
- Theoretically not too hard to compute area flooded when rise to h meters
 - But no software can do it for Denmark at 2-meter resolution
- Use of I/O-efficient algorithms
⇒ Denmark in a few days
- Even compute new terrain where terrain below h is flooded when water rise to h



Interdisciplinary Collaboration

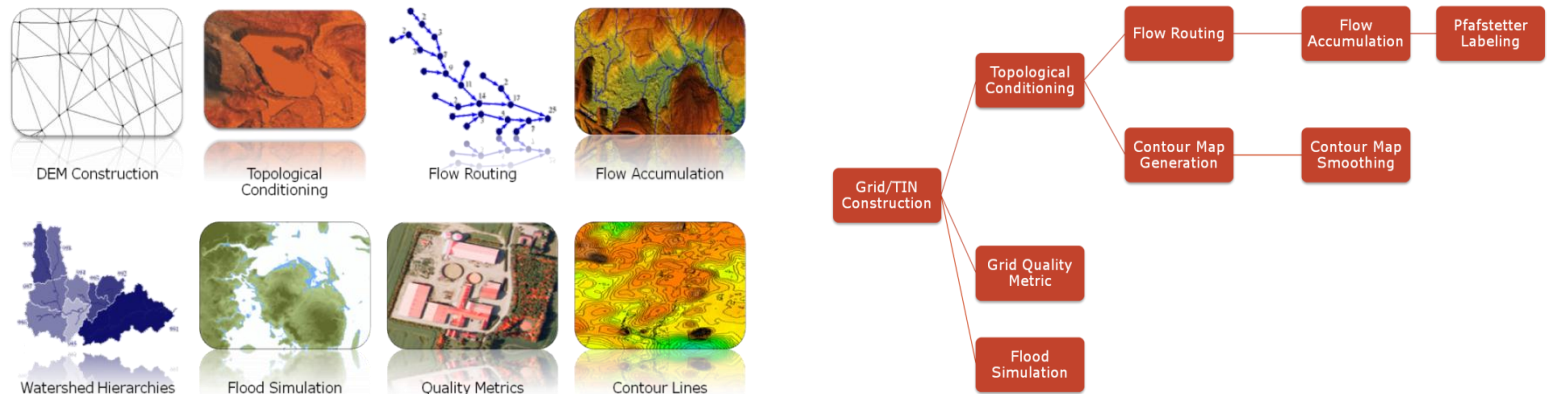
- Flow/flooding work example of center interdisciplinary collaboration
 - Flood modeling and efficient algorithms in biodiversity modeling
 - Allow for used of global and/or detailed geographic data

- Brings together researchers from
 - Biodiversity
 - Ecoinformatics
 - Algorithms
 - Datamining
 - ...



TerraSTREAM

- Flow/flooding work part of comprehensive software package
 - **TerraSTREAM**: Whole pipeline of terrain data processing software



- TerraSTREAM used on full 2 meter Denmark model
(~25 billion points, ~1.5 TB)
 - Terrain model (grid) from LIDAR point data
 - Surface flow modeling: Flow directions and flow accumulation
 - Flood modeling

Summary

- Basic research center in Aarhus
 - Organization
 - PhD education
- Examples of research
 - Theoretical external memory algorithmics
 - Practical (flow simulation)

Tau · Jërë-jëf · Tashakkur · S.aHHa · Sag olun
Giihtu · Djakujo · Dâkujem vám · Thank you
Tesekkür ederim · To-siä · Merci · Tashakur
Taing · Dankon · Efharisto´ · Shukriya · Kiitos
Dhanyabad · Rakhmat · Trugarez · Asante
Köszönöm · Blagodarya · Dziekuje · Eskerrik asko
Grazie · Tak · Bayarlaya · Miigwech · Dank u
Spasibo · Dêkuji vám · Ngiyabonga · Dziakuj
Obrigado · Gracias · A dank aych · Salamat
Takk · Arigatou · Tack · Tänan · Aciu
Korp kun kah · Multumesk · Terima kasih · Danke
Rahmat · Gratias · Mahalo · Dhanyavaad
Paldies · Faleminderit · Diolch · Hvala
Kam-sa-ham-ni-da · Xiè xiè · Mèrcie · Dankie

Gerth Stølting Brodal

gerth@cs.au.dk