DATALOGISK INSTITUT
DET NATURVIDENSKABELIGE FAKULTET
AARHUS UNIVERSITET

Udfordringer ved håndtering af massive datamængder:

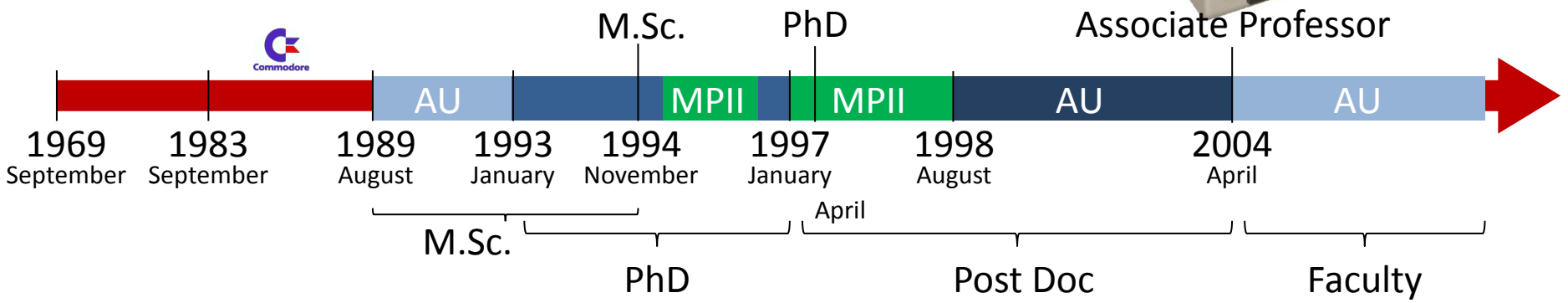
Forskingen ved Grundforskningscenteret for Massive Data Algortihmics

Gerth Stølting Brodal

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Universitets-Samvirket Århus, Statsbiblioteket, Århus, November 16, 2010

Gerth Stølting Brodal



Outline of Talk

- MADALGO
  CENTER FOR MASSIVE DATA ALGORITHMICS
  - who, where, what ?
  - reseach areas
- External memory algorithmics
  - models
  - searching and sorting
- Flow simulation
- Fault-tolerant searching


MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

MADALGO

CENTER FOR MASSIVE DATA ALGORITHMICS

– Where?

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

- Center of Danmarks Grundforskningsfond Danish National Research Foundation
- **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

MADALGO

CENTER FOR MASSIVE DATA ALGORITHMICS

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

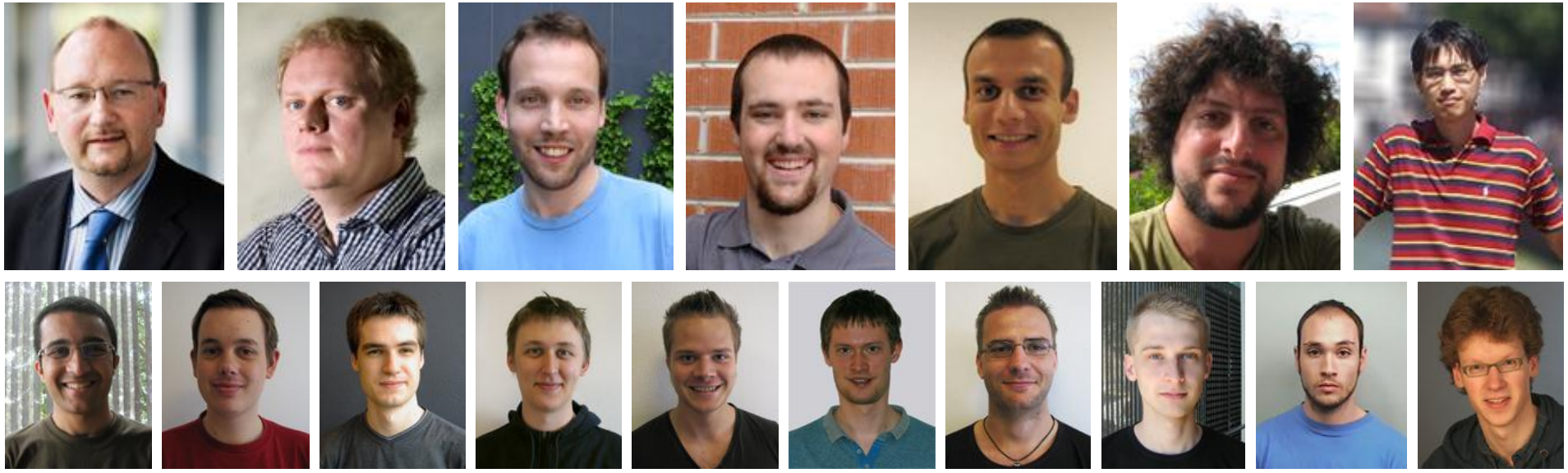
Jakob Truelsen

Kostas Tsakalidis

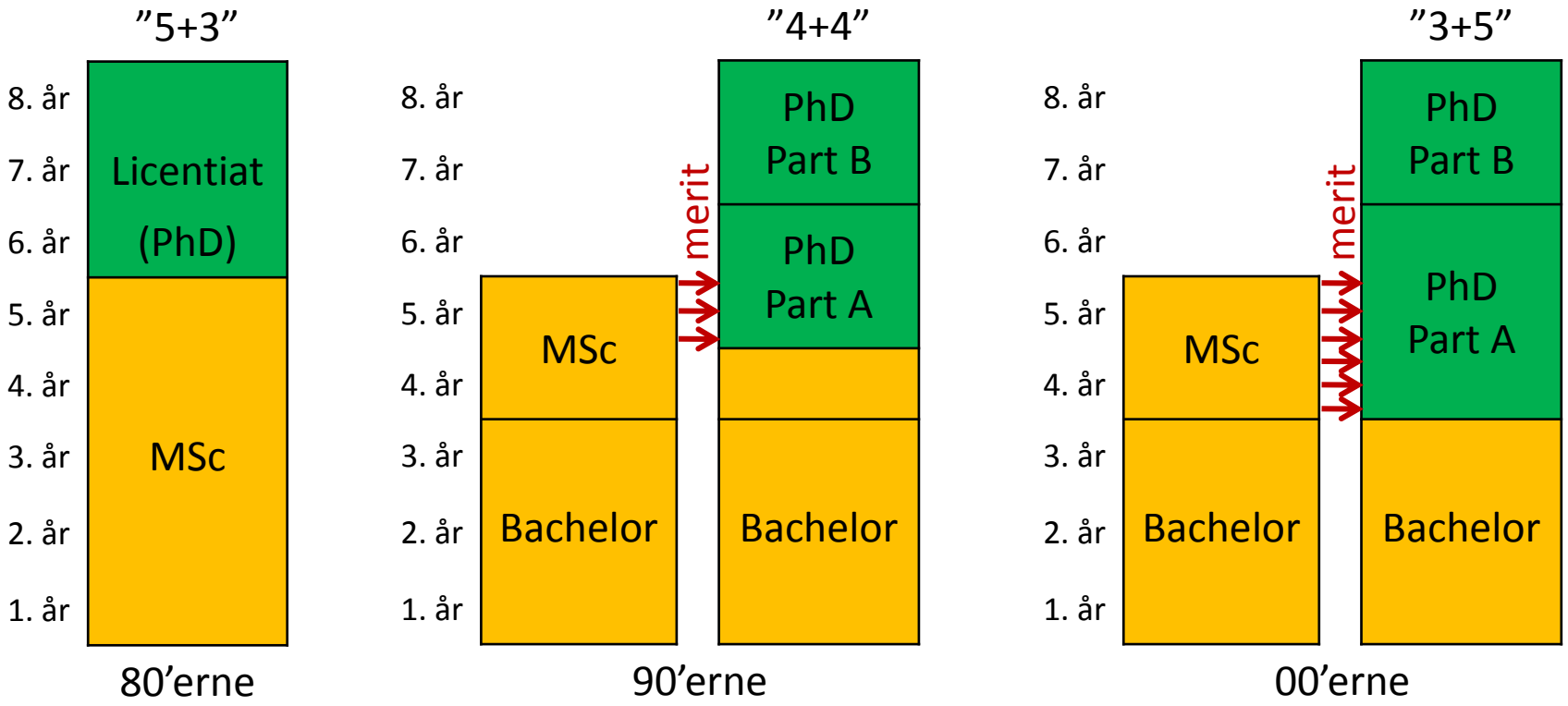
Mark Greve

Morten Revsbæk

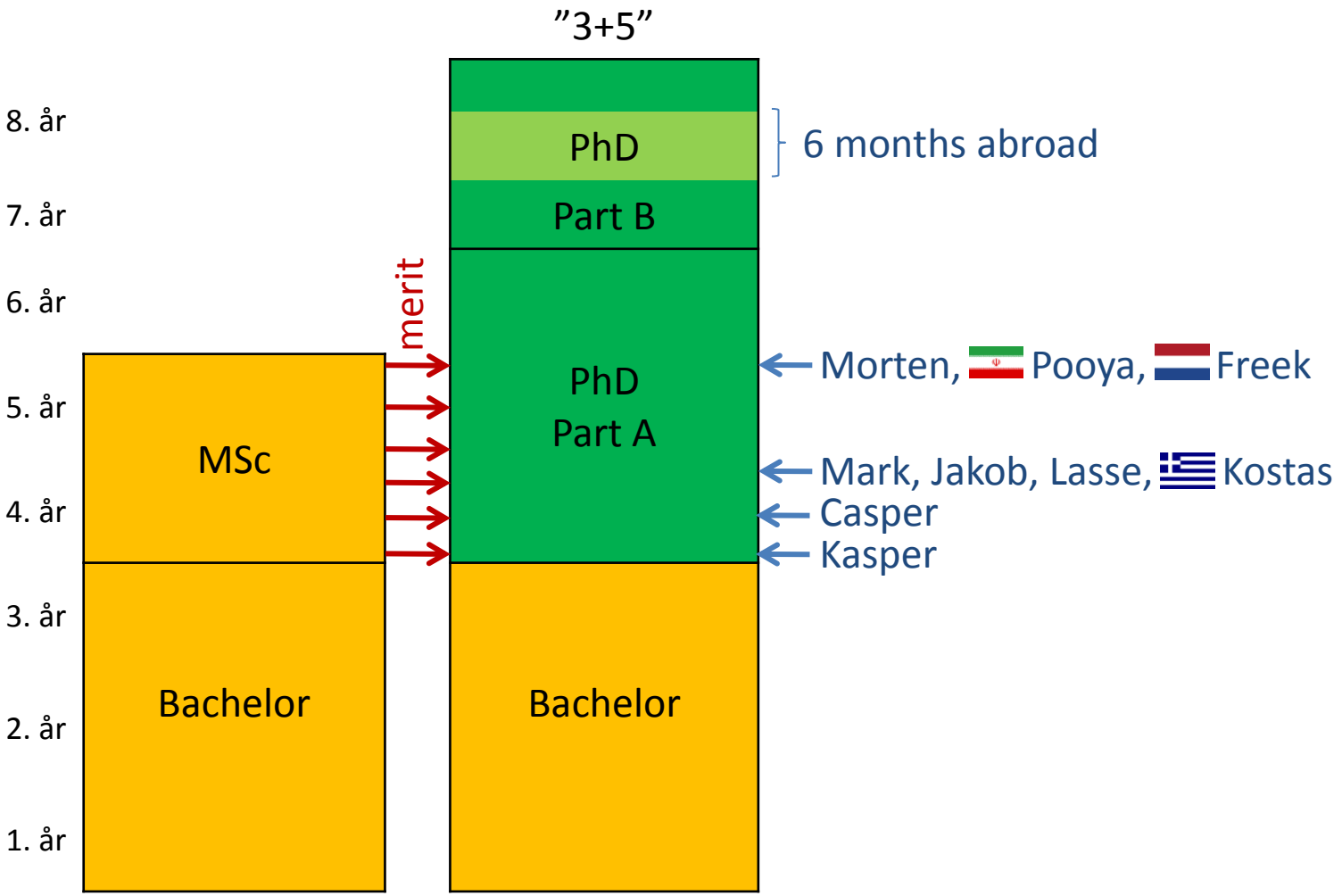
Pooya Davoodi



PhD Education @ AU



PhD Education @ MADALGO




snowbird

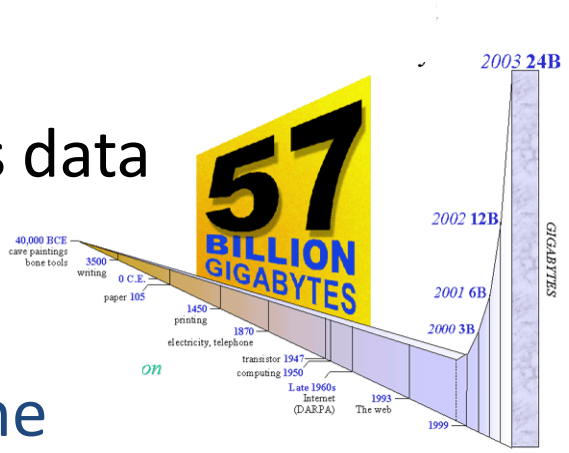

MASSIVE 2010
Summer School 2010
SoCG 2009
Retreat 2009
Massive Data Algorithmics
Retreat 2008
MASSIVE 2009
Summer School 2008
Retreat 2007
Summer School 2007
Inauguration 2007

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

- *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



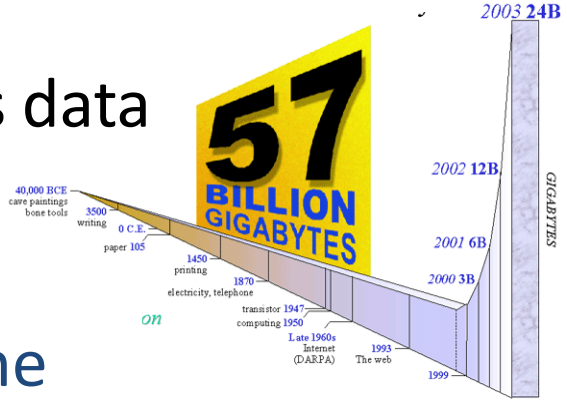
Nature special issues

- 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

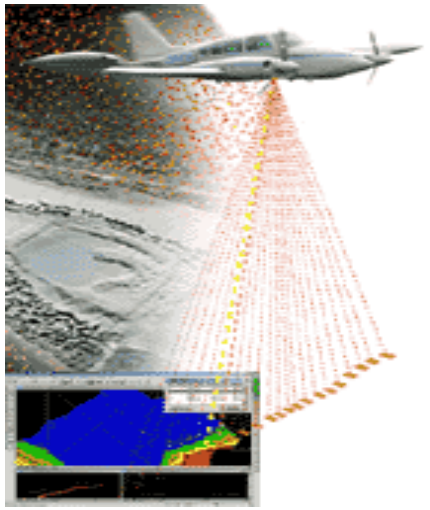


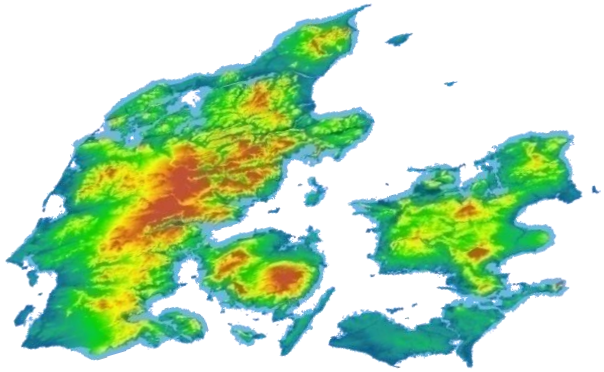
Obviously not only in sciences:

- 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)





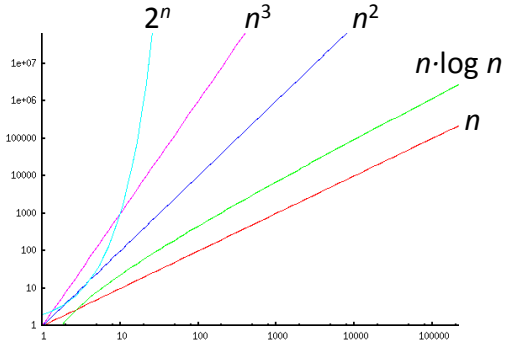
Denmark

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

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

Algorithm Inadequacy

- **Algorithms**: Problem solving “recipies”



- Importance of **scalability/efficiency**
  → Algorithmics core computer science area

- Traditional algorithmics:
  Transform input to output using simple machine model

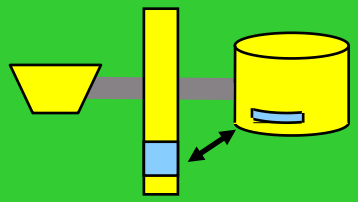
- Inadequate with e.g.
  - Massive data
  - Small/diverse devices
  - Continually arriving data

  → Software inadequacies!

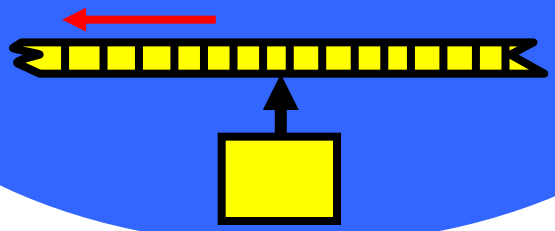
MADALGO

CENTER FOR MASSIVE DATA ALGORITHMICS

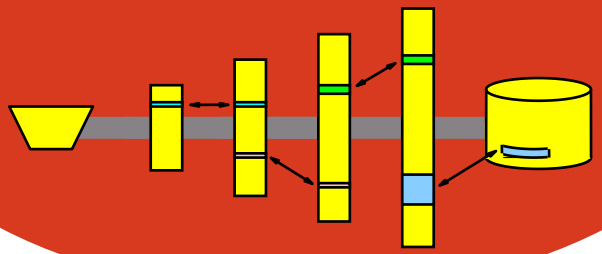
I/O Efficient Algorithms



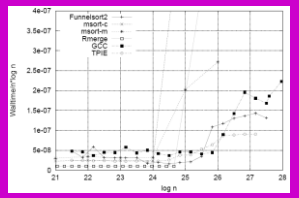
Streaming Algorithms

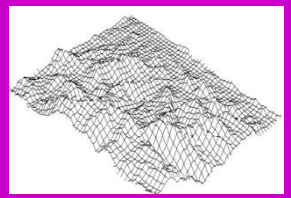


Cache Oblivious Algorithms

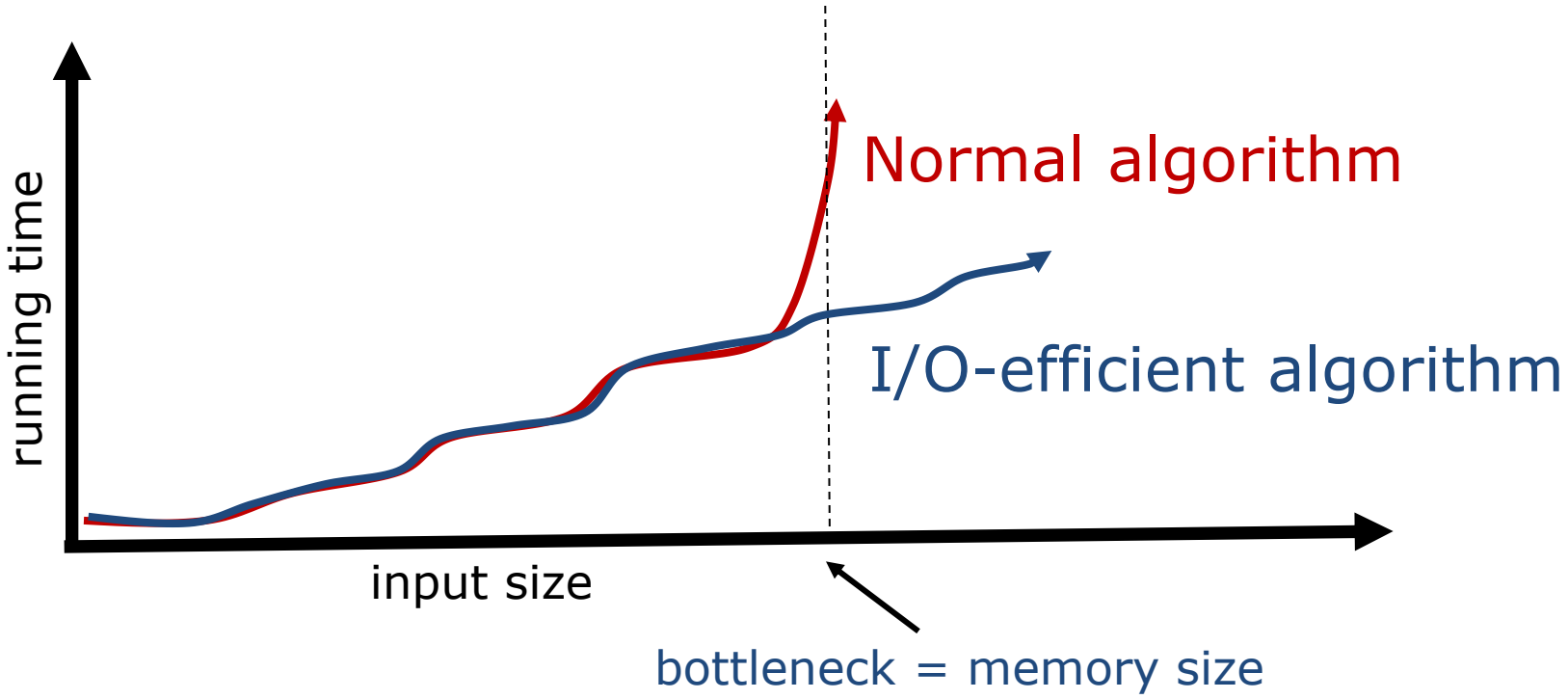


Algorithm Engineering

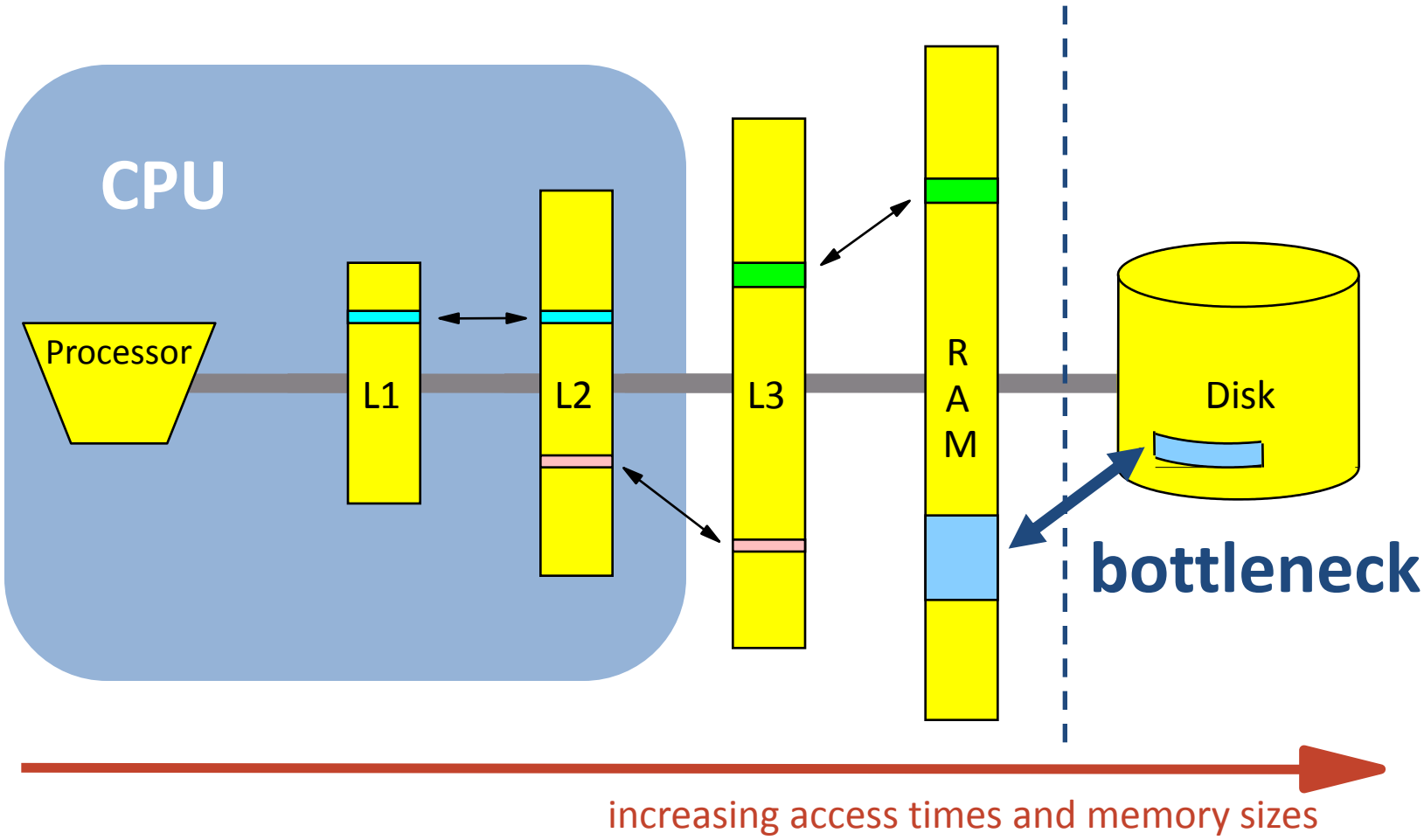




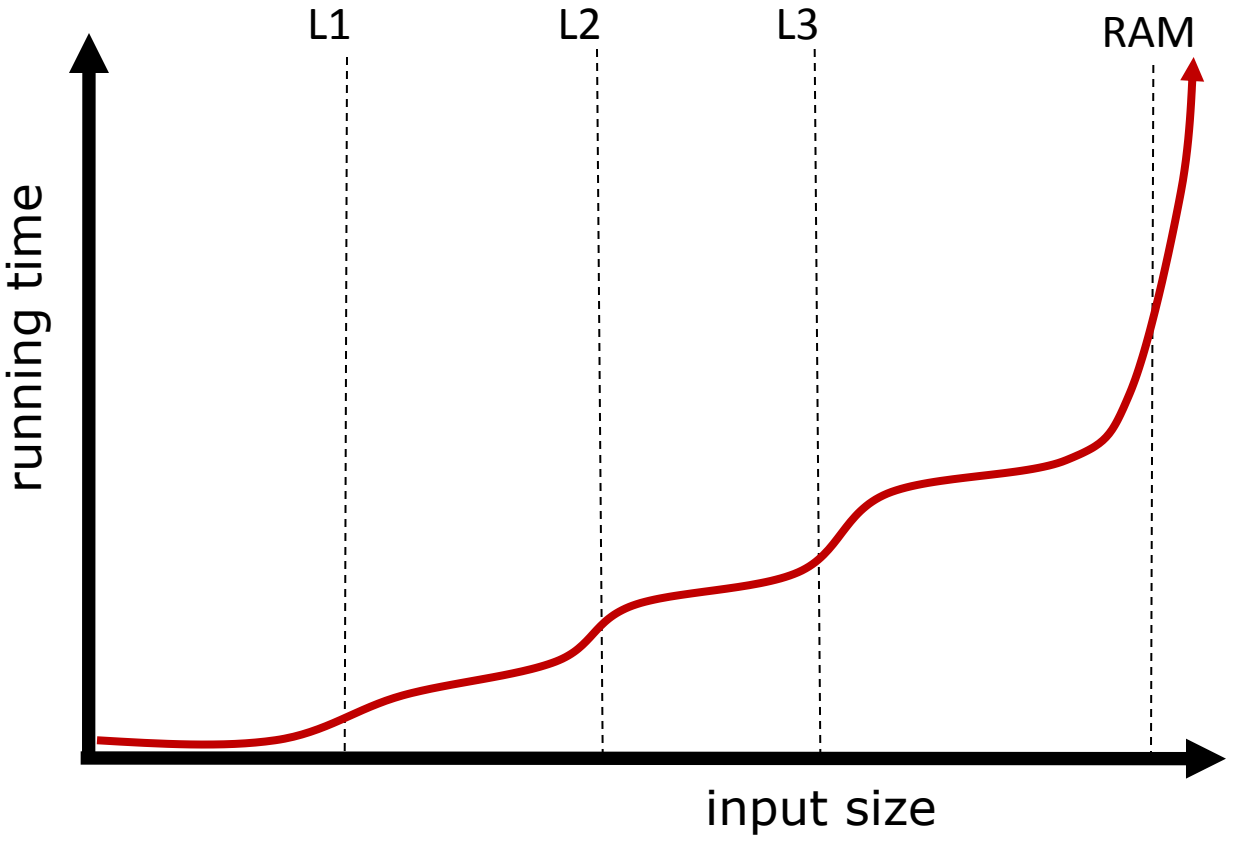
The problem...


running time
input size
Normal algorithm
I/O-efficient algorithm
bottleneck = memory size

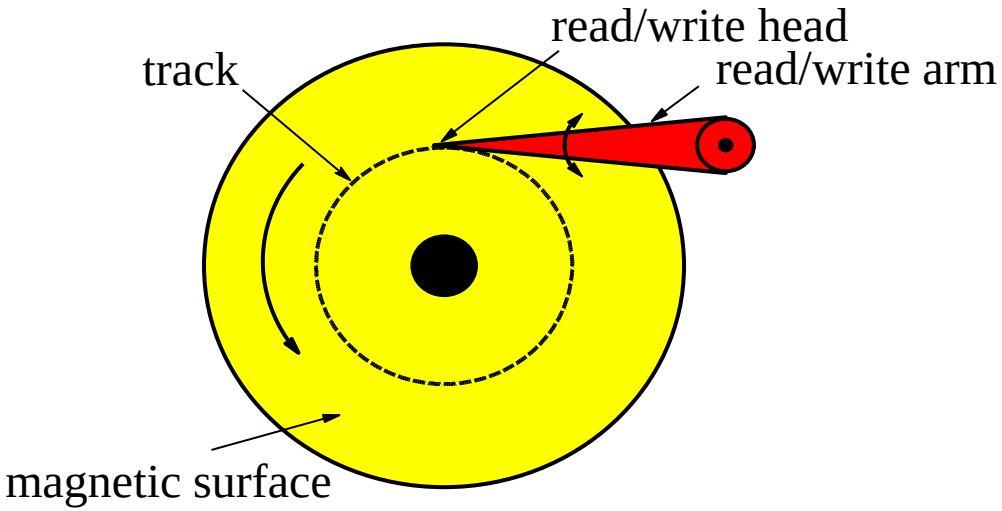
Memory Hierarchies



Memory Hierarkies vs. Running Time

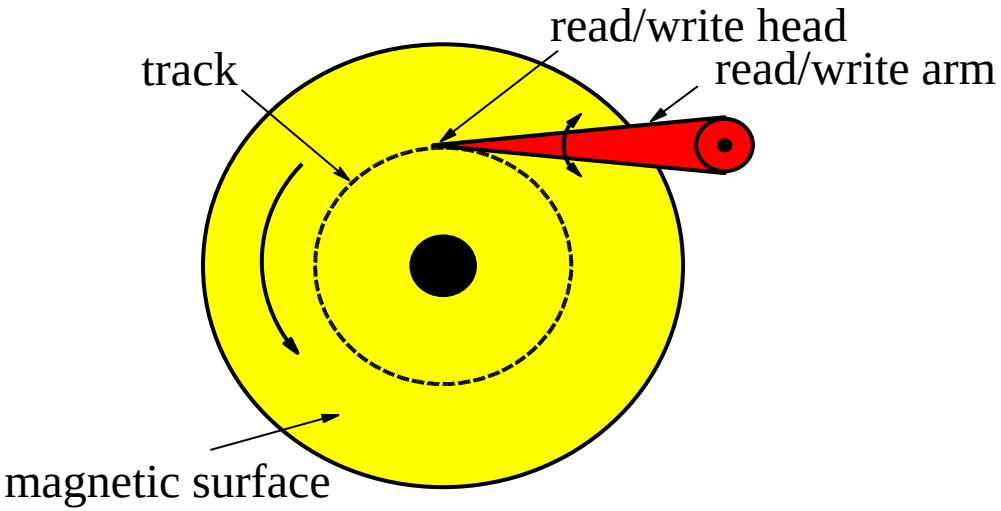


Disk Mechanics



"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 !

I/O-efficient algorithms

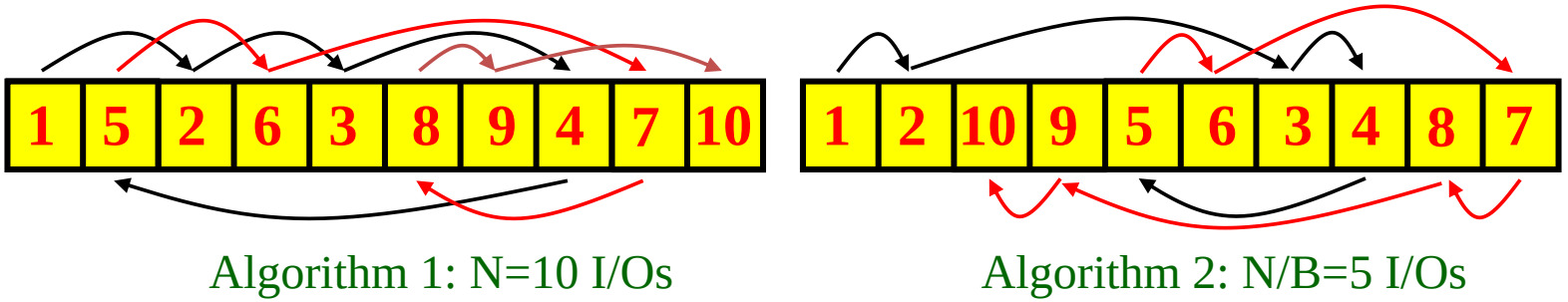
Move as few disk blocks as possible to solve given problem *!*

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)



Algorithm 1: N=10 I/Os

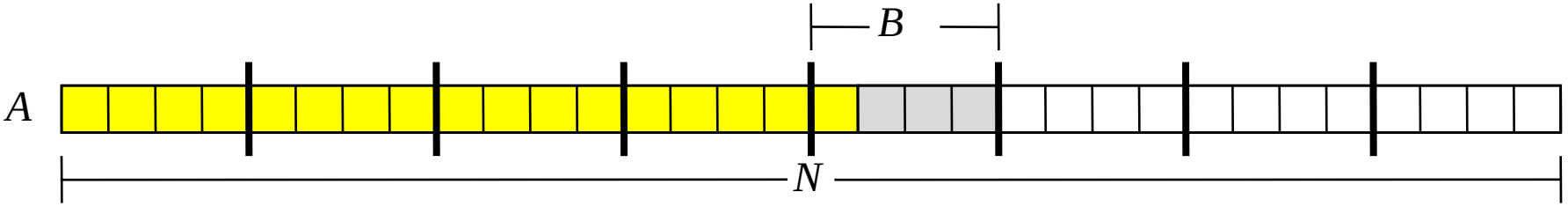
Algorithm 2: N/B=5 I/Os

- 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

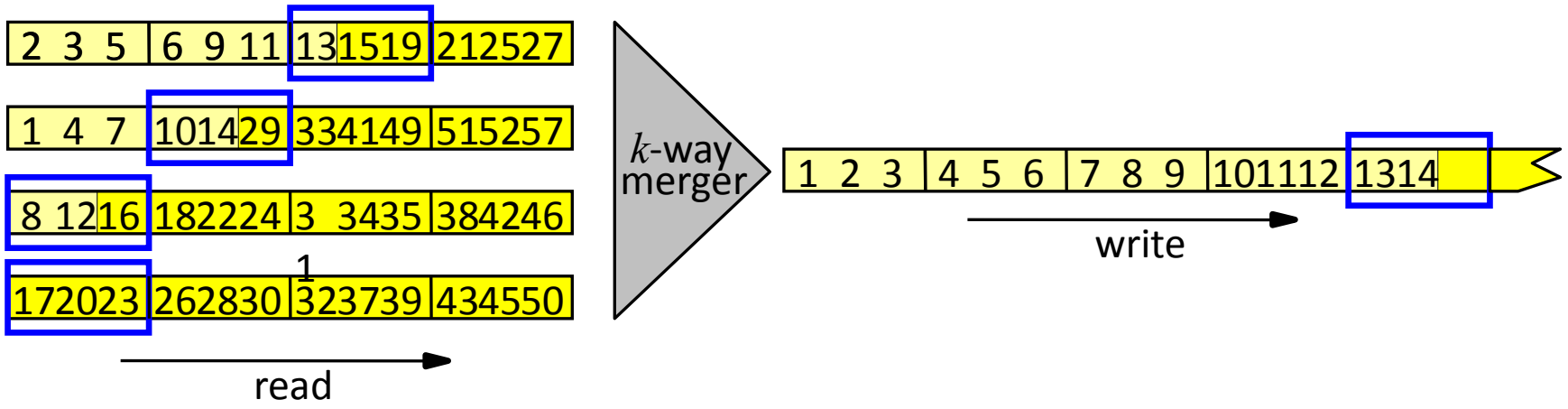
    $\Rightarrow$ N/B I/Os take $256/8$ $sec = 32$ sec

I/O Efficient Scanning



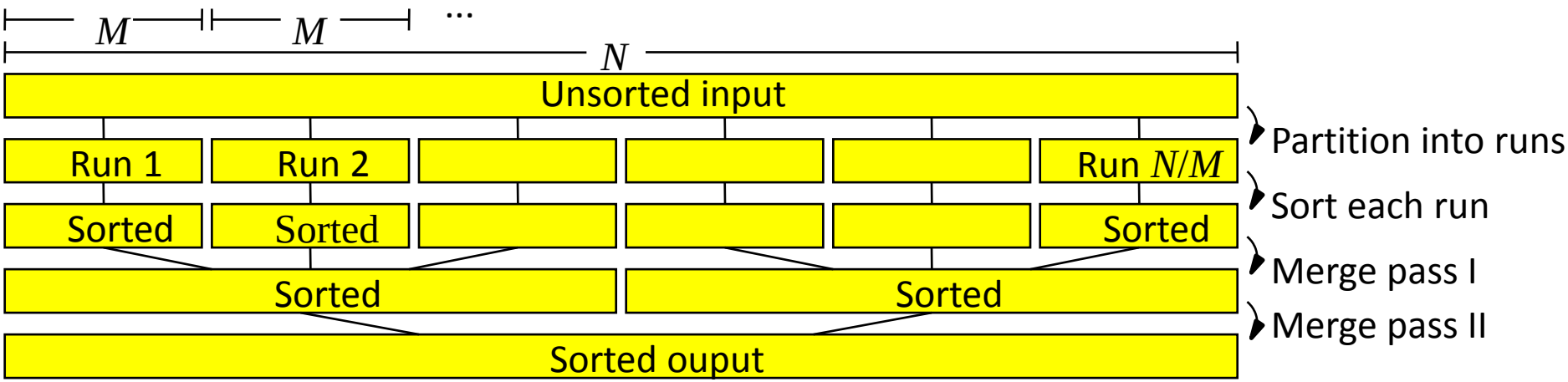
$O(N/B)$ I/Os

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

Energy-Efficient Sorting using Solid State Disks

(Bechman, Meyer, Sanders, Siegler 2010)

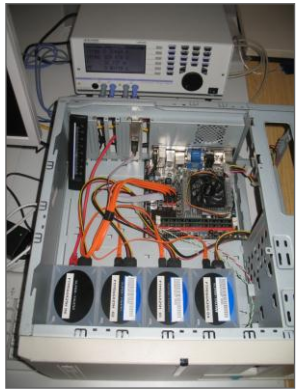
Sorting large data sets

- Is easily described
- Has many applications
- Stresses both CPU and the I/O system
- Benchmark introduced 1985

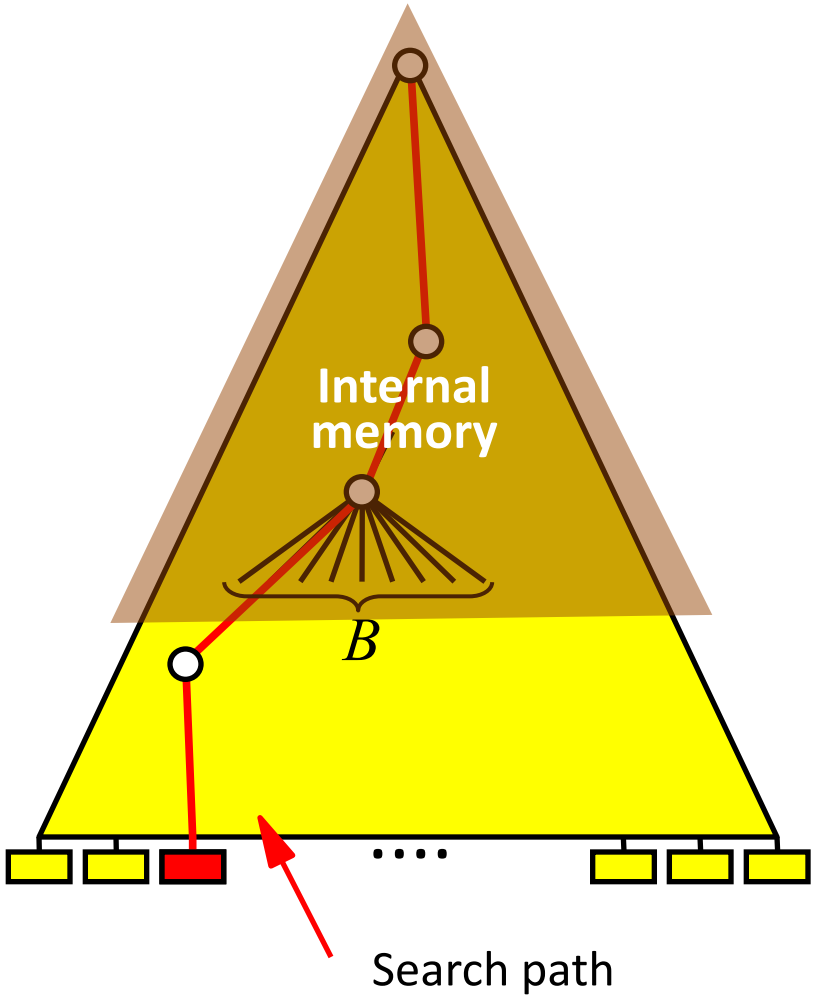
Energy Efficiency

- Energy (and cooling) is a significant cost factor in data centers
- Energy consumption correlates to pollution

	2007			2010			
Size [GB]	Time [s]	Energy [kJ]	Rec./J	Time [s]	Energy [kJ]	Rec./J	Energy Saving Factor
10	86.6	8.6	11628	76.7	2.8	35453	3.0
100	881	88.1	11354	756	27.5	36381	3.2
1000	7196*	2920*	3425	21906	723.7	13818	4.0



B-trees -
The Basic Searching Structure



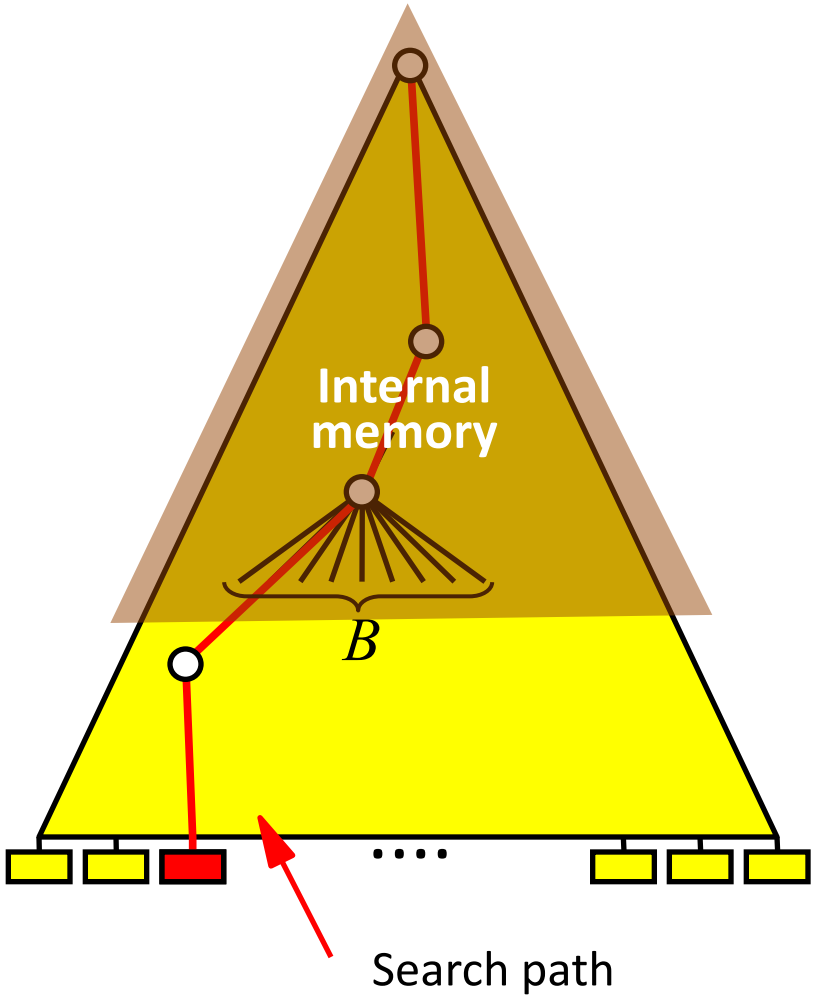
- **Searches**

  Practice: 4-5 I/Os

- **Repeated searching**

  Practice: 1-2 I/Os

B-trees

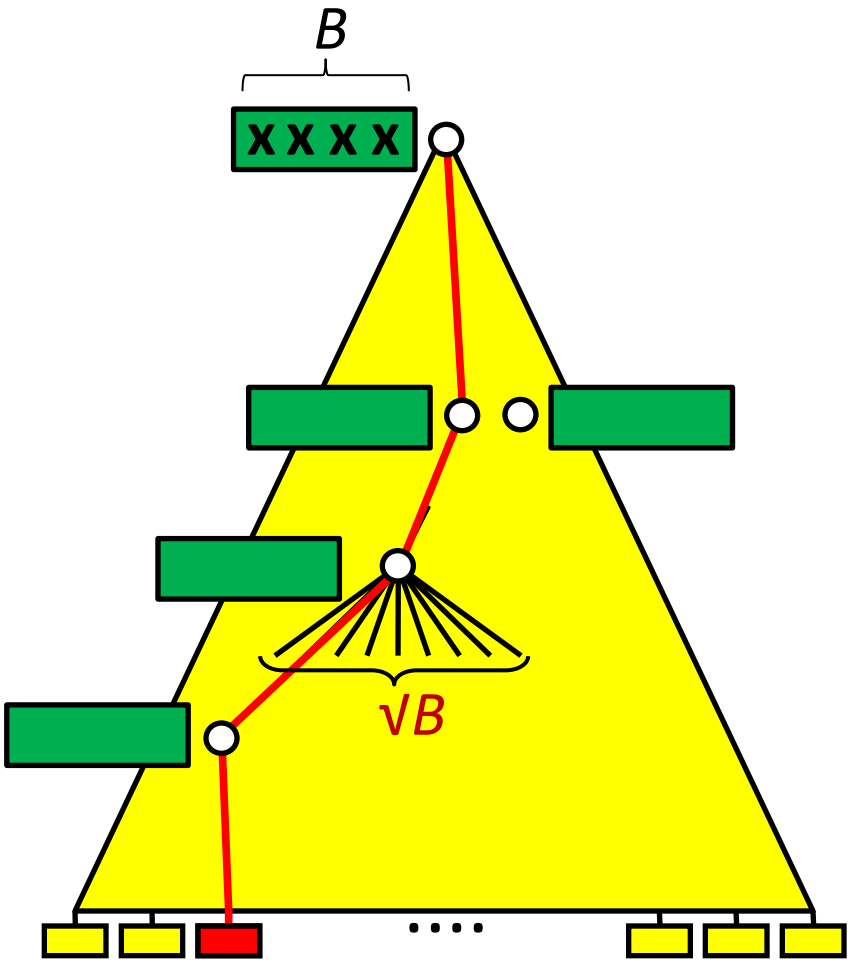


Best possible

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

?

B-trees with Buffered Updates



- **Searches cost**

  $O(\log_B N)$ I/Os

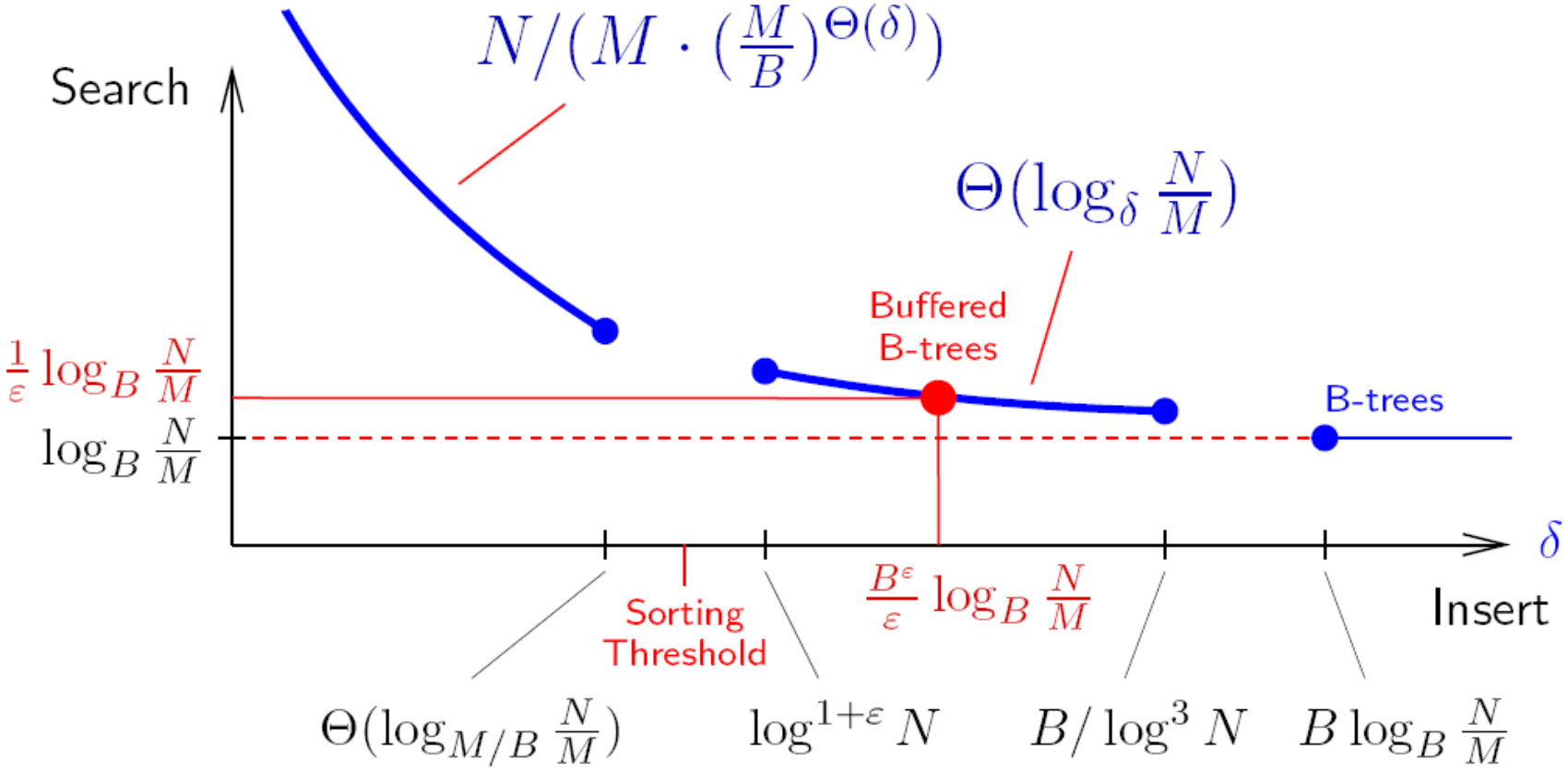
- ***N* updates cost**

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

Trade-off between search and update times – optimal *!*

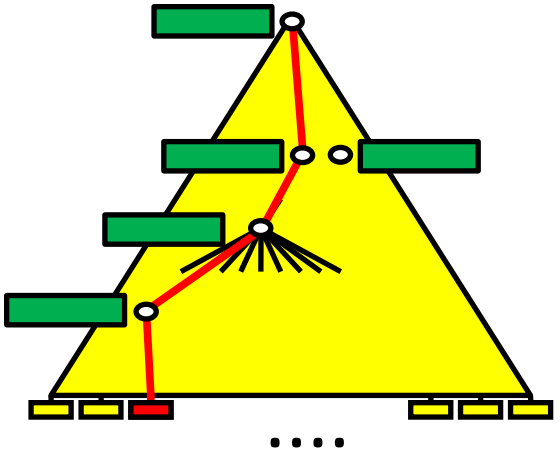
Brodal and Fagerberg (2003)

B-trees with Buffered Updates



B-trees with Buffered Updates Experimental Study

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



Flood Prediction Important

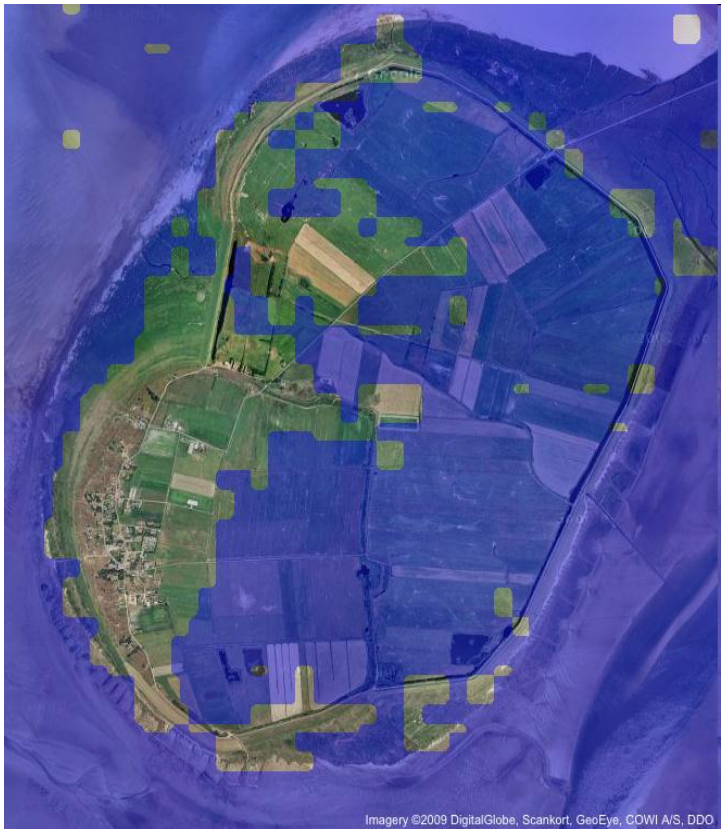
- Prediction areas susceptible to floods
  - Due to e.g. raising sea level or heavy rainfall
- Example: Hurricane Floyd Sep. 15, 1999

7 am

3pm

Detailed Terrain Data Essential

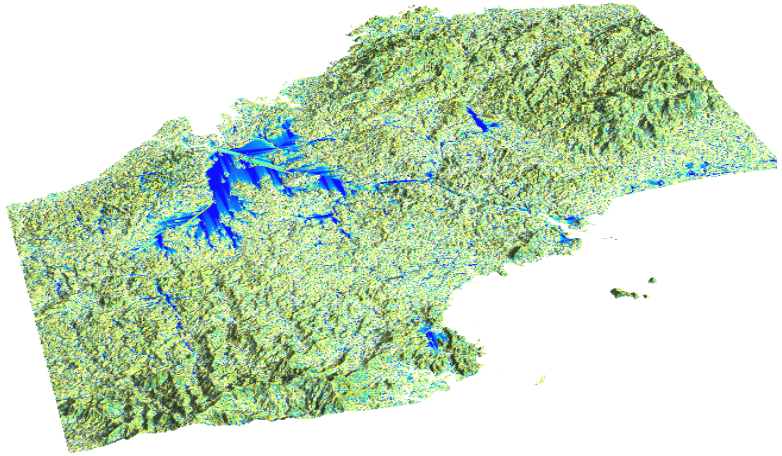
Mandø with 2 meter sea-level raise



80 meter terrain model

2 meter terrain model

Surface Flow Modeling

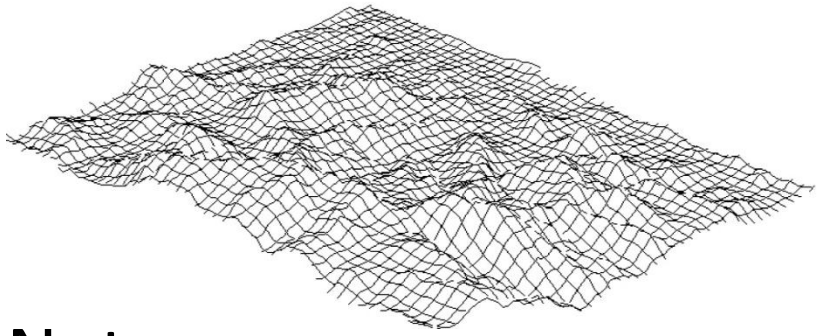


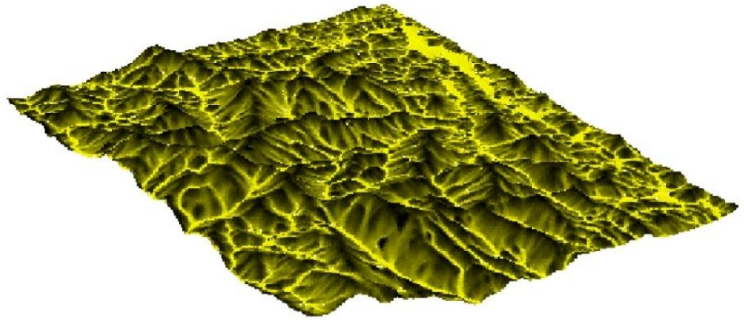
Hurricane Floyd (September 15, 1999)

- 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

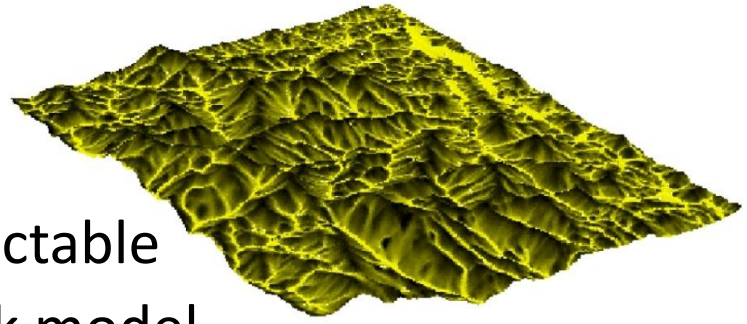
- Flow accumulation on grid terrain model:
  - Initially one unit of water in each grid cell
  - Water (initial and received) distributed from each cell to lowest lower neighbor cell (if existing)
  - Flow accumulation of cell is total flow through it





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

Massive Data Problems



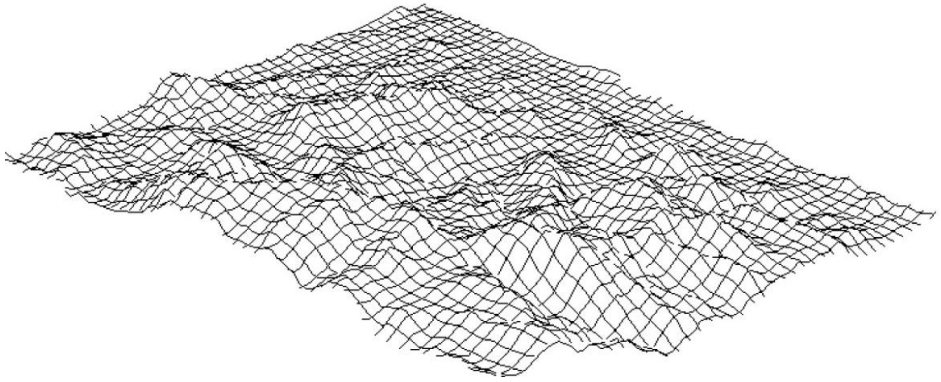
- Commercial systems:
  - Often *very* slow
  - Performance somewhat unpredictable
  - Cannot handle 2-meter Denmark model

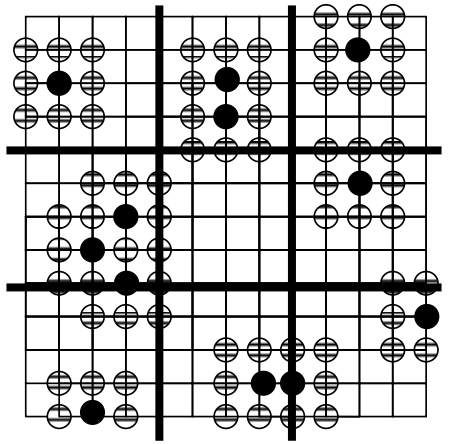
- Collaboration environmental researchers in late 90'ties
  - US Appalachian mountains dataset
    - 800x800km at 100m resolution ⇒ a few Gigabytes 14 days!!
    - Customized software on ½ GB machine:

- Appalachian dataset would be Terabytes sized at 1m resolution!

Flow Accumulation Performance

- Natural algorithm access disk for each grid cell
  - "Push" flow down the terrain by visiting cells in height order
    $\Rightarrow$ Problem since cells of same height scattered over terrain

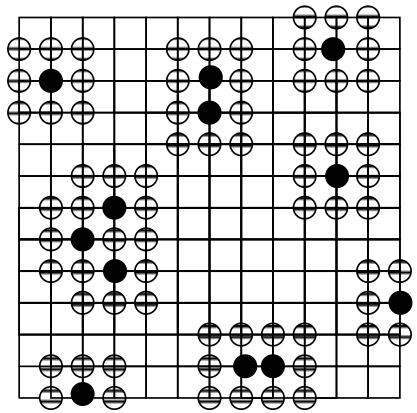




- Natural to try "tiling" (commercial systems?)
  - But computation in different tiles not independent

I/O-Efficient Flow Accumulation

- We developed I/O-optimal algorithms (assessing disk a lot less)
- Avoiding scattered access by:
  - Grid storing input: Data duplication
  - Grid storing flow: “Lazy write”



- Implementation very efficient
  - Appalachian Mountains flow accumulation in 3 hours!
  - Denmark at 2-meter resolution in a few days

Flood Modeling

- Not all terrain below height *h* is flooded when water rise to *h* meters!

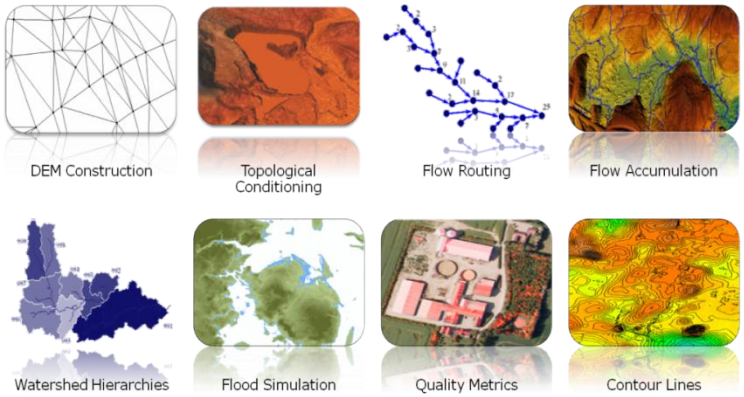
- Theoretically not too hard to compute area flooded when rise to *h* meters
  - But no software could do it for Denmark at 2-meter resolution

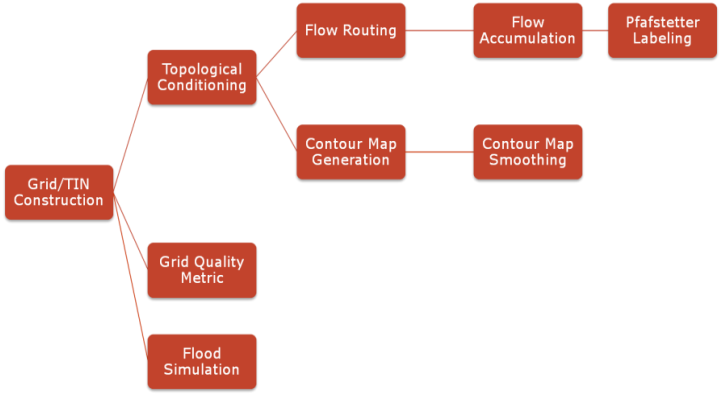
- Use of I/O-efficient algorithms
  $\Rightarrow$ Denmark in a few days

- Even compute new terrain where terrain below *h* is flooded when water rise to *h*

TerraSTREAM

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





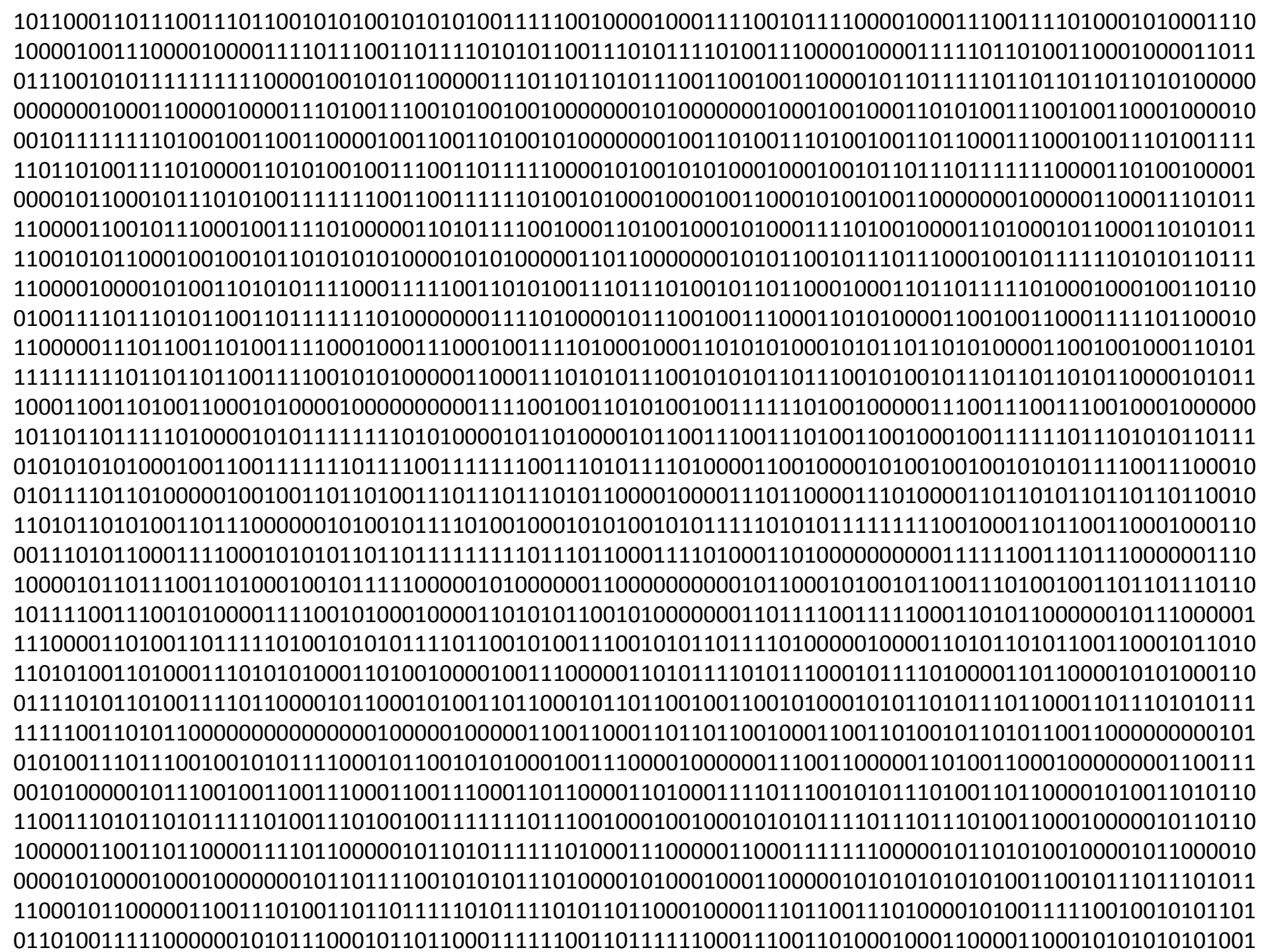
- TerraSTREAM used on full 2 meter Denmark model

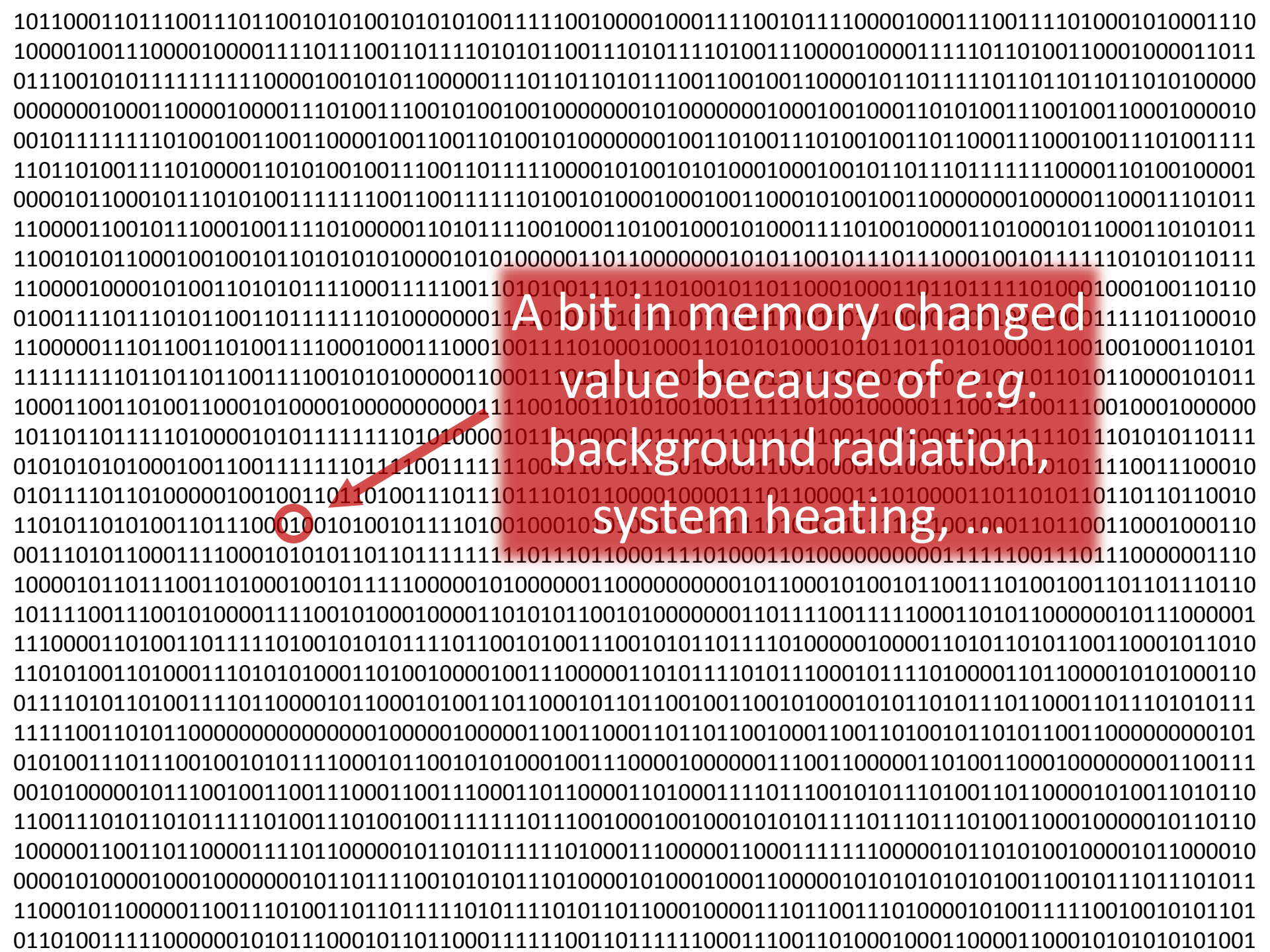
  (~ 25 billion points, ~ 1.5 TB)
  - Terrain model (grid) from LIDAR point data
  - Surface flow modeling: Flow directions and flow accumulation
  - Flood modeling

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
  - …

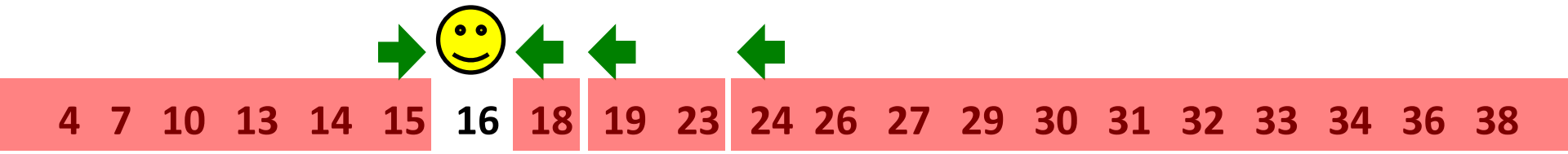



A bit in memory changed value because of *e.g.* background radiation, system heating, ...

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

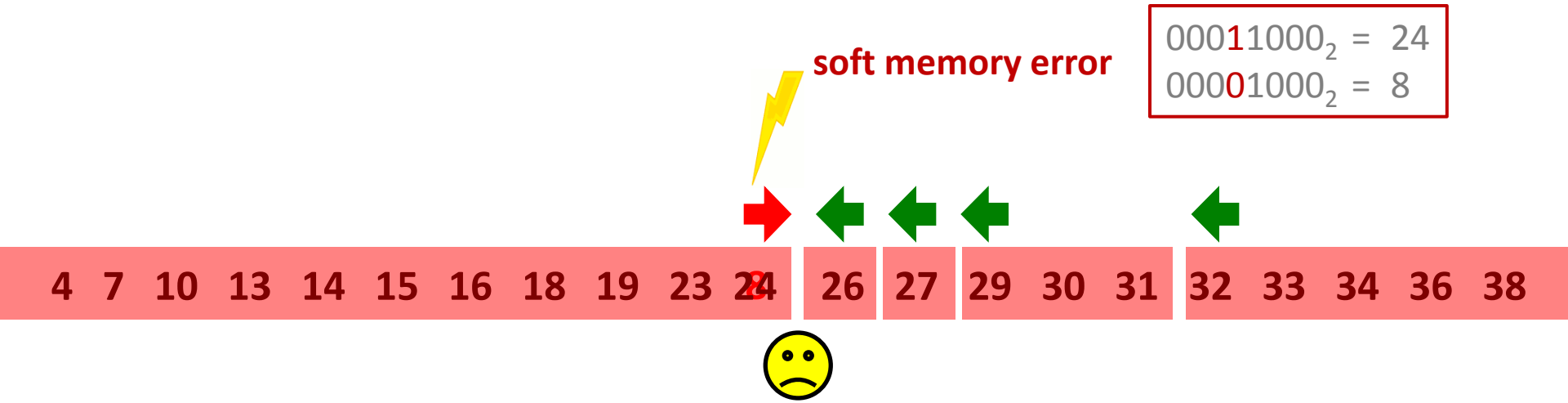
— Google fellow Jeff Dean

Binary Search for 16



$O(\log N)$ comparisons

Binary Search for 16



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

Where is Laurits ?

Where is Laurits ?

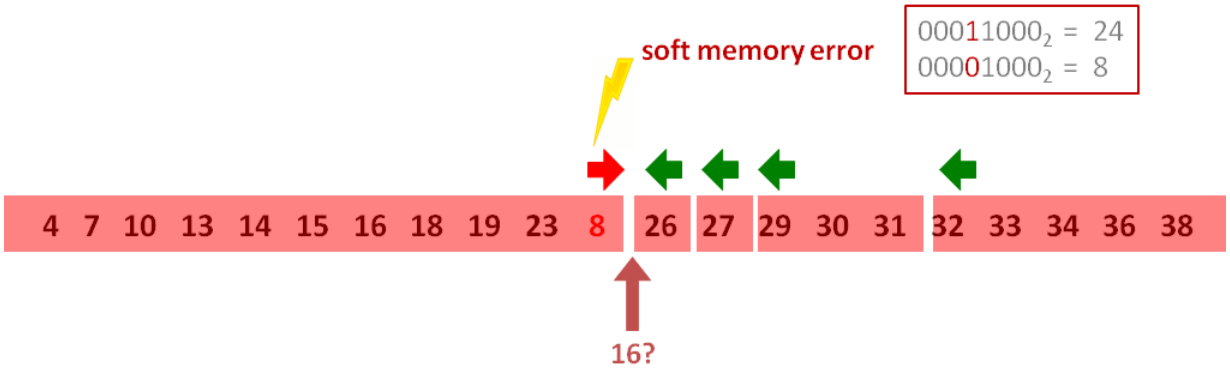
Where is Laurits ?

If at most 4 faulty answers then Laurits is somewhere here

Faulty-Memory RAM Model

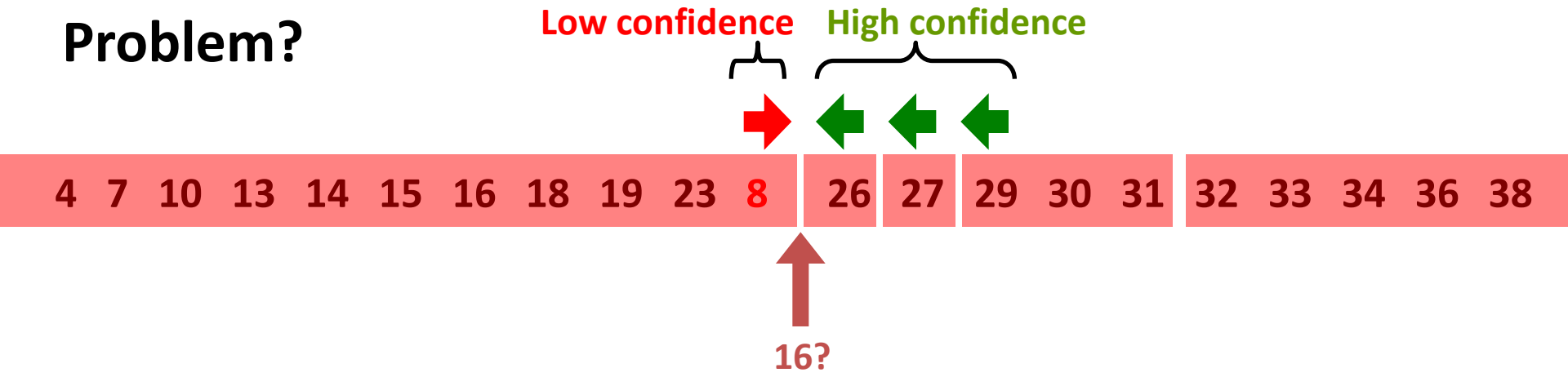
Finocchi and Italiano, STOC'04

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



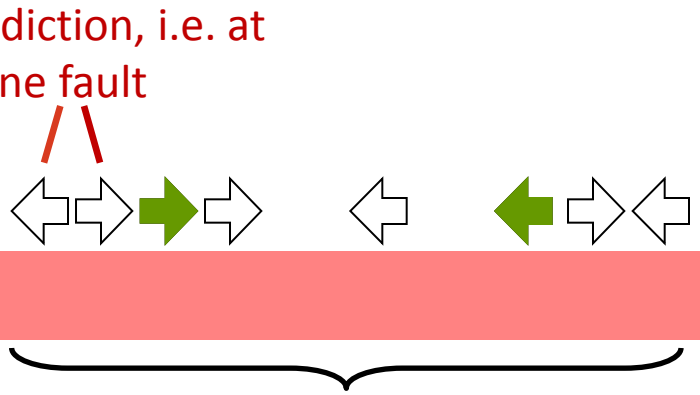
Faulty-Memory RAM: Searching

Problem?



Faulty-Memory RAM: Searching

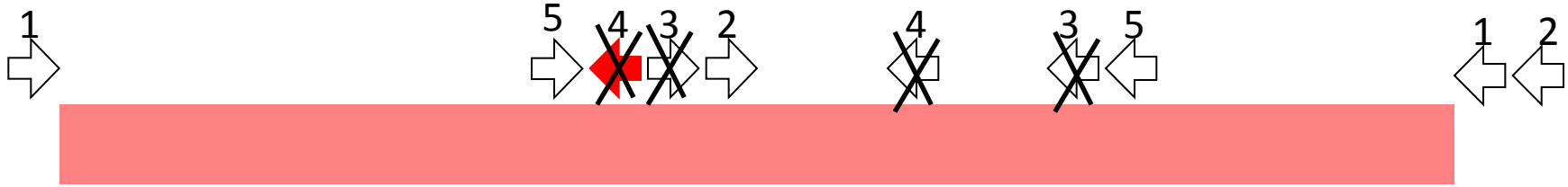
When are we done (δ=3)?



If range contains at least δ+1 ⇨ and δ+1 ⇦ then there is at least one uncorrupted ➡ and ⬅, i.e. *x* must be contained in the range

Faulty-Memory RAM: Θ(log *N* + δ) 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

Summary

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

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Tau · Jërë-jëf · Tashakkur · S.aHHa · Sag olun
Giihtu · Djakujo · Dâkujem vám · Thank you
Tesekkür ederim · To-siä · Merci · Tashakur
Taing · Dankon · Efharistó · Shukriya · Kiitos
Dhanyabad · Rakhmat · Trugarez · Asante
Köszönöm · Blagodarya · Dziekuje · Eskerrik asko
Grazie · Tak · Bayarlaa · 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 · Xìe xìe · Mèrcie · Dankie

Thank You

Gerth Stølting Brodal
gerth@cs.au.dk