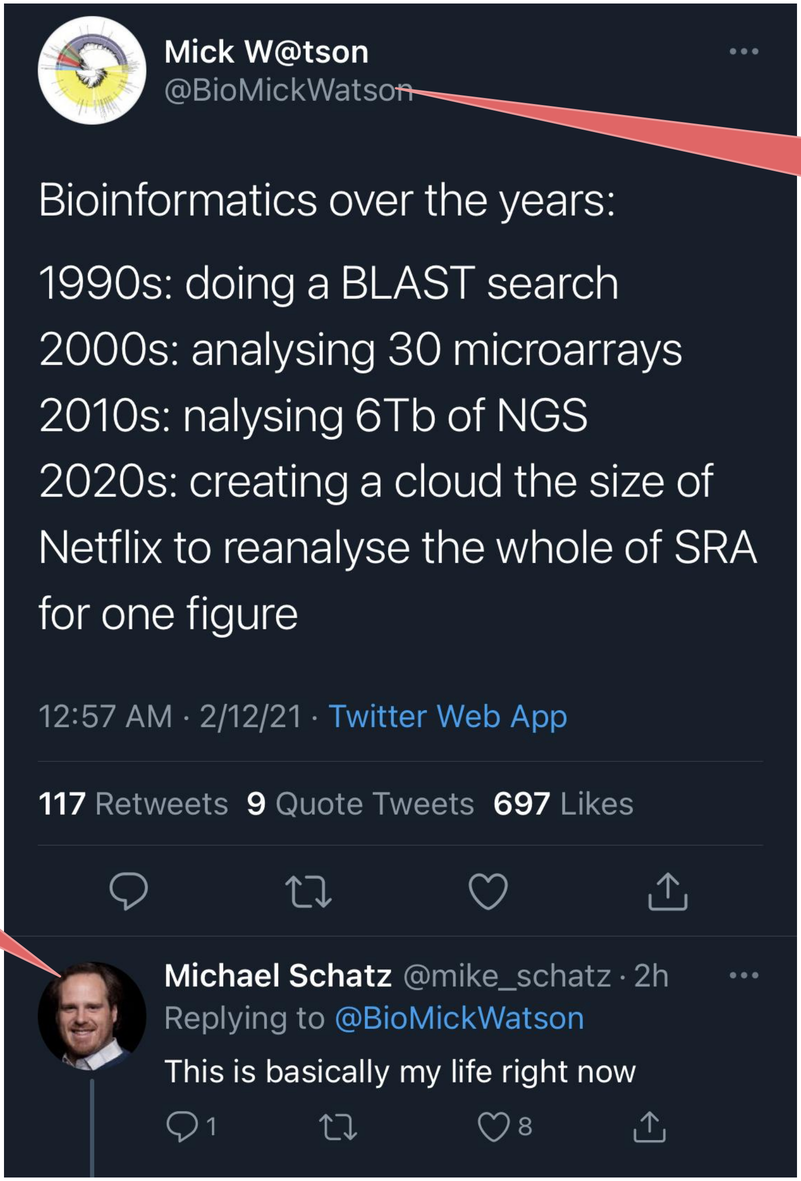
Data Systems at Scale:
Scaling Up by Scaling Down and Out

Prashant Pandey
ppandey@berkeley.edu
Berkeley Lab/UC Berkeley

Bioinformatics on twitter



Mick W@tson
@BioMickWatson

Bioinformatics over the years:

1990s: doing a BLAST search
2000s: analysing 30 microarrays
2010s: nalysing 6Tb of NGS
2020s: creating a cloud the size of Netflix to reanalyse the whole of SRA for one figure

12:57 AM · 2/12/21 · Twitter Web App

117 Retweets **9** Quote Tweets **697** Likes

Michael Schatz @mike_schatz · 2h
Replying to @BioMickWatson
This is basically my life right now

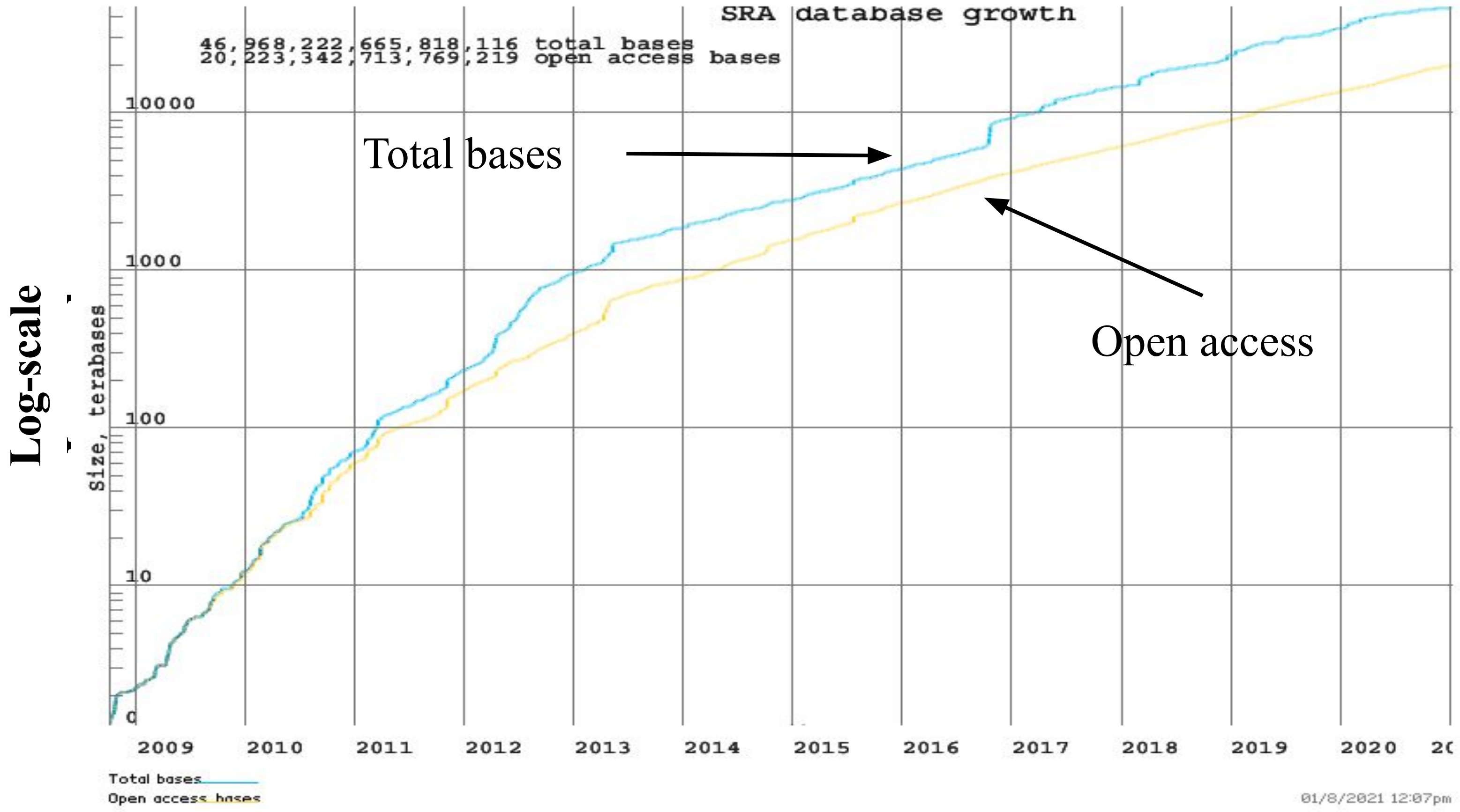
1 8

Professor Bioinformatics and Computational Biology
The University of Edinburgh

Associate Professor Computational Biology
Johns Hopkins University

Sequence Read Archive (SRA) growth

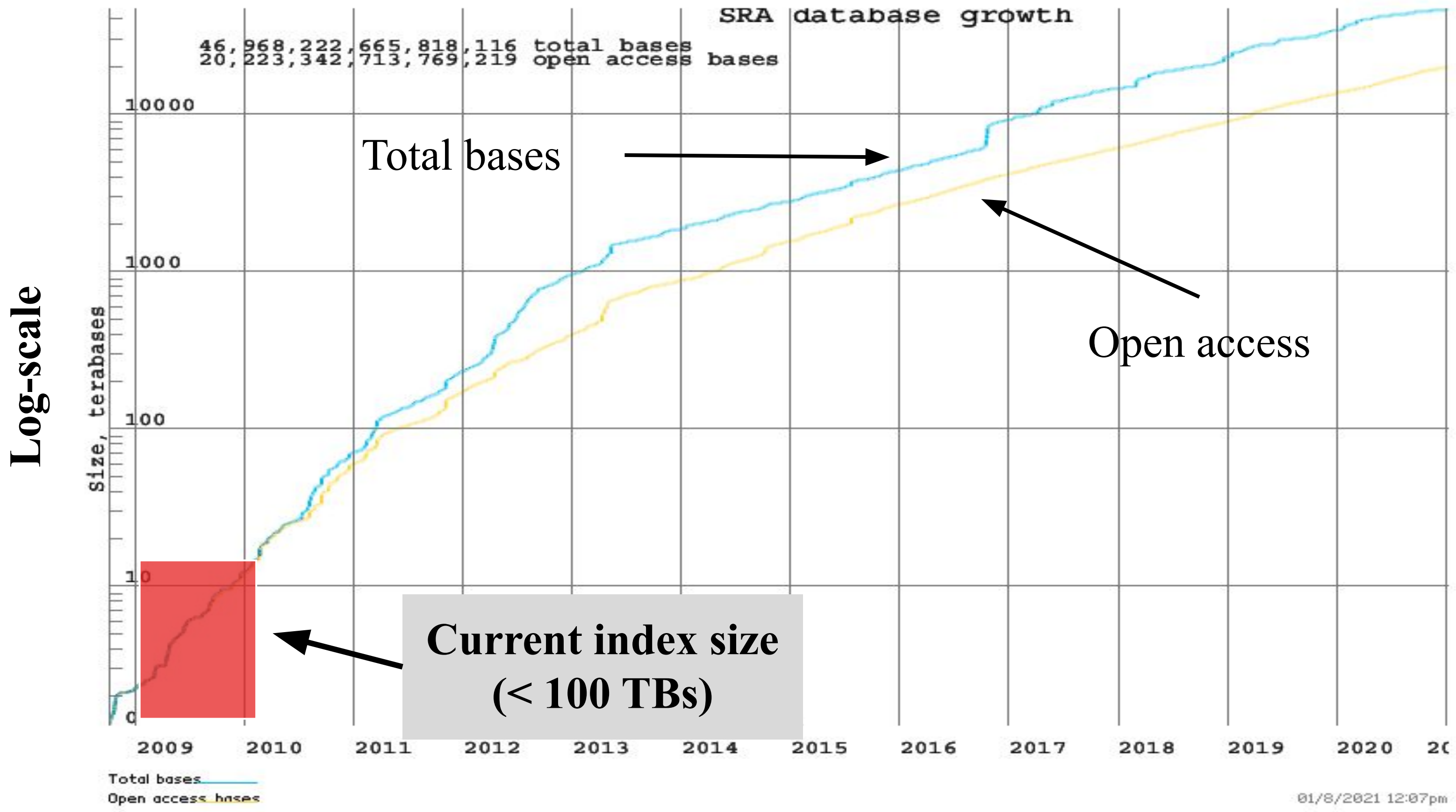
SRA contains a lot of ***diversity information***



Q: What if I find e.g., a new disease-related gene, and want to see if it appeared in other experiments?

Scalability is the bottleneck for data science

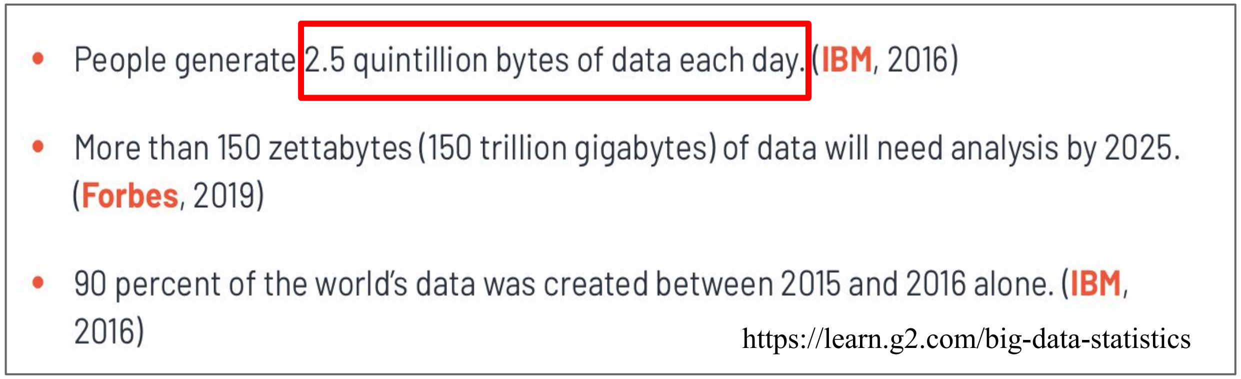
SRA contains a lot of ***diversity information***



This renders what is otherwise an immensely valuable public resource ***largely inert***

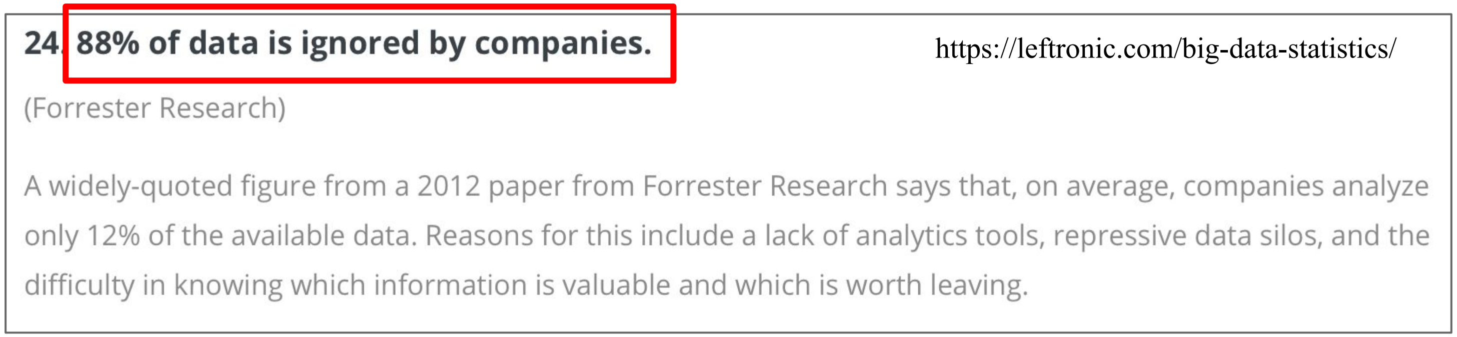
Scalability is a ubiquitous challenge

Cyber monitoring
Internet of Things
Financial tech
Social networks
AstroPhysics
Chemistry
Environmental science
•
•
•
•
•



- People generate 2.5 quintillion bytes of data each day. (IBM, 2016)
- More than 150 zettabytes (150 trillion gigabytes) of data will need analysis by 2025. (Forbes, 2019)
- 90 percent of the world's data was created between 2015 and 2016 alone. (IBM, 2016)

https://learn.g2.com/big-data-statistics



24. 88% of data is ignored by companies.

(Forrester Research)

A widely-quoted figure from a 2012 paper from Forrester Research says that, on average, companies analyze only 12% of the available data. Reasons for this include a lack of analytics tools, repressive data silos, and the difficulty in knowing which information is valuable and which is worth leaving.

https://leftronic.com/big-data-statistics/

My goal as a researcher

My goal as a researcher is to build ***scalable data systems*** to ***accelerate*** and ***scale*** next generation data analysis

Three approaches to handle massive data

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:

- Compact & succinct data structures
- Filters, e.g., Bloom, quotient, etc.

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:

- Compact & succinct data structures
- Filters, e.g., Bloom, quotient, etc.

Organize it

Goal: organize data in a disk-friendly way

Techniques:

- B-tree
- B^{ε}-tree
- LSM-tree

Three approaches to handle massive data

Shrink it

Goal: make data smaller to fit in RAM

Techniques:
- Compact & succinct data structures
- Filters, e.g., Bloom, quotient, etc.

Organize it

Goal: organize data in a disk-friendly way

Techniques:
- B-tree
- B^{ε}-tree
- LSM-tree

Distribute it

Goal: partition and distribute data on multiple nodes

Techniques:
- Distributed hash table
- Distributed key-value store

Research output

Shrink	Organize	Distribute

Data structures & Algorithms

Systems

Applications

Computational biology

Research output

🟥 Shrink | 🟩 Organize | 🟦 Distribute

Data structures & Algorithms

🟥 **Quotient Filter** SIGMOD '17, SIGMOD '21

🟩 **Buffered CMS** ESA '18, **Scalable MG** arXiv '19

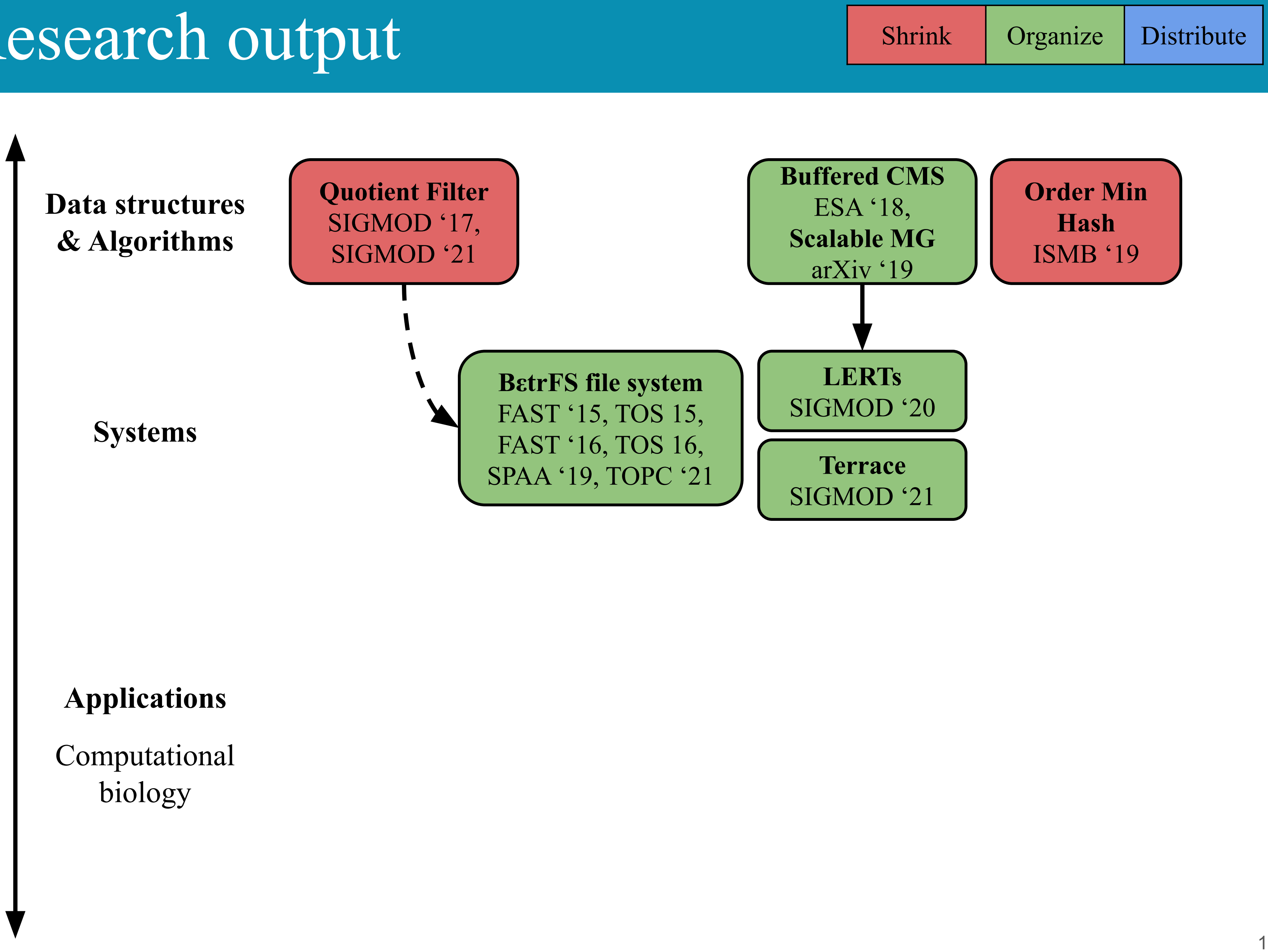
🟥 **Order Min Hash** ISMB '19

Systems

Applications

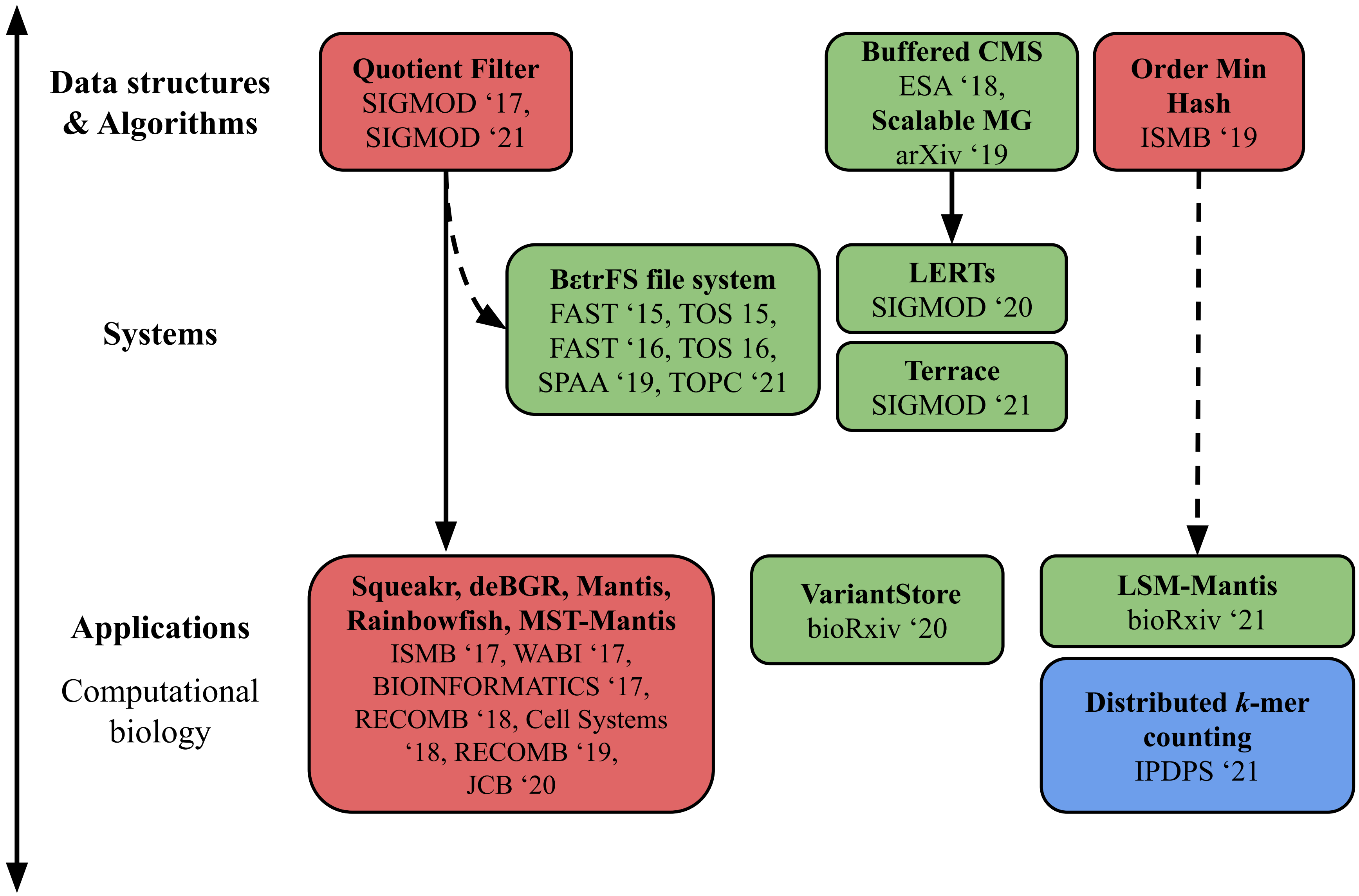
Computational biology

Research output


Shrink
Organize
Distribute
Data structures & Algorithms
Systems
Applications
Computational biology
Quotient Filter SIGMOD '17, SIGMOD '21
Buffered CMS ESA '18, Scalable MG arXiv '19
Order Min Hash ISMB '19
BεtrFS file system FAST '15, TOS 15, FAST '16, TOS 16, SPAA '19, TOPC '21
LERTs SIGMOD '20
Terrace SIGMOD '21

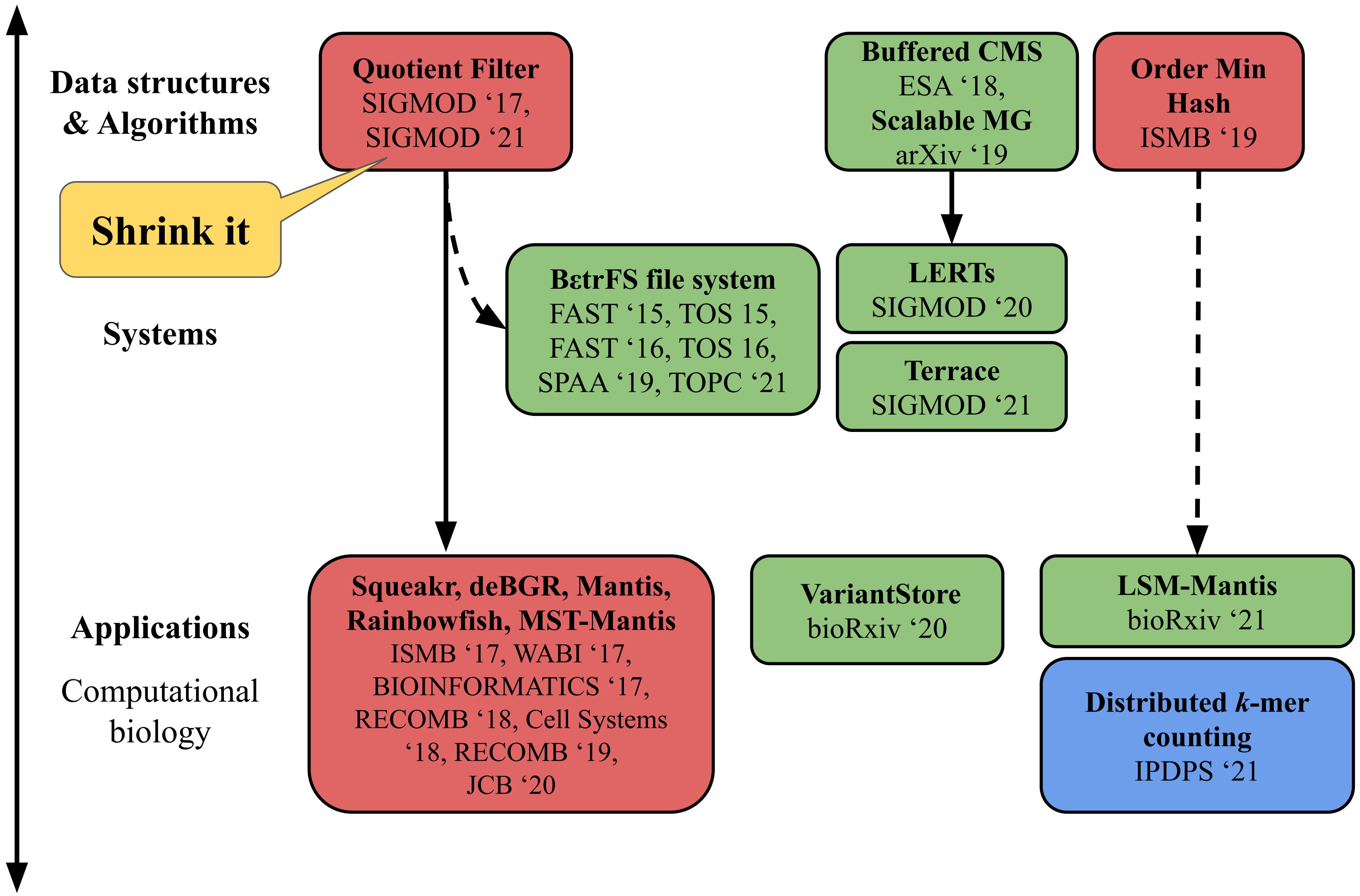
Research output

🟥 Shrink 🟩 Organize 🟦 Distribute


Data structures & Algorithms
Systems
Applications
Computational biology
Quotient Filter SIGMOD '17, SIGMOD '21
Buffered CMS ESA '18, Scalable MG arXiv '19
Order Min Hash ISMB '19
BεtrFS file system FAST '15, TOS 15, FAST '16, TOS 16, SPAA '19, TOPC '21
LERTs SIGMOD '20
Terrace SIGMOD '21
Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20
VariantStore bioRxiv '20
LSM-Mantis bioRxiv '21
Distributed k-mer counting IPDPS '21

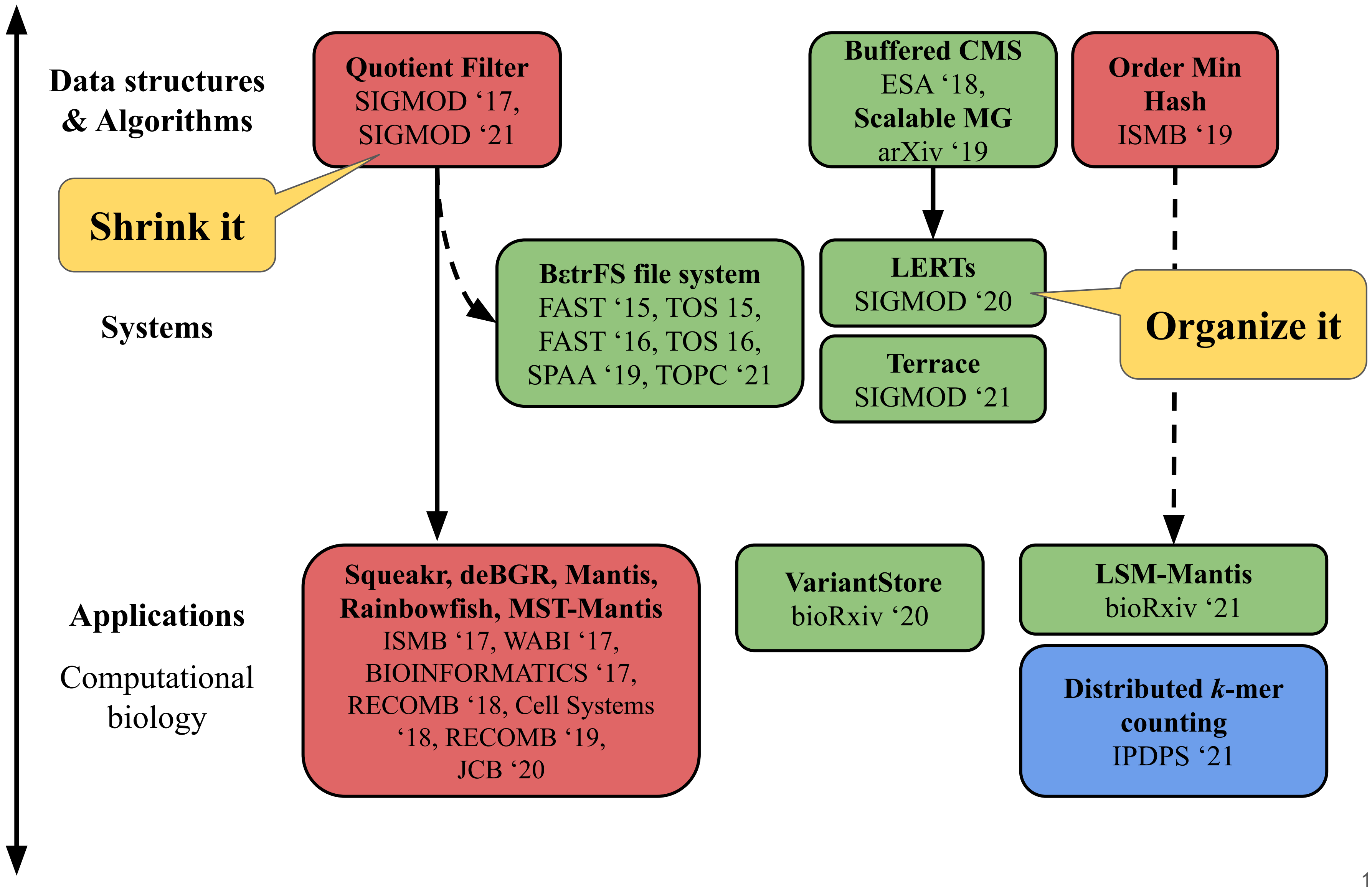
In this talk:

🟥 Shrink | 🟩 Organize | 🟦 Distribute


Data structures & Algorithms
Systems
Applications
Computational biology
Shrink it
Quotient Filter SIGMOD '17, SIGMOD '21
Buffered CMS ESA '18, Scalable MG arXiv '19
Order Min Hash ISMB '19
BεtrFS file system FAST '15, TOS 15, FAST '16, TOS 16, SPAA '19, TOPC '21
LERTs SIGMOD '20
Terrace SIGMOD '21
Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20
VariantStore bioRxiv '20
LSM-Mantis bioRxiv '21
Distributed k-mer counting IPDPS '21

In this talk:

Shrink | Organize | Distribute

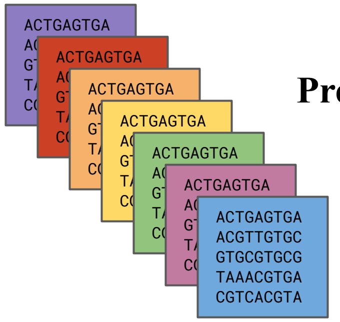

Data structures & Algorithms
Systems
Applications
Computational biology
Shrink it
Organize it
Quotient Filter SIGMOD '17, SIGMOD '21
Buffered CMS ESA '18, **Scalable MG** arXiv '19
Order Min Hash ISMB '19
BεtrFS file system FAST '15, TOS 15, FAST '16, TOS 16, SPAA '19, TOPC '21
LERTs SIGMOD '20
Terrace SIGMOD '21
Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis ISMB '17, WABI '17, BIOINFORMATICS '17, RECOMB '18, Cell Systems '18, RECOMB '19, JCB '20
VariantStore bioRxiv '20
LSM-Mantis bioRxiv '21
Distributed *k*-mer counting IPDPS '21

Computational biology applications using quotient filters

Application 1: *k*-mer counting

Biological samples

Sequencing



Raw sequencing data (Sequence Read Archive)

Preprocessing

***k*-mer**	**Count**
ACTGATG	10
GGTGCAT	20
AACTGGA	2
CCGTGAC	1000
GGTGTGC	4000000
CGTGCAC	11
GTGTCAC	9090992

***k*-mer counting > 30 papers**

1. Genome/metagenome assembly
2. Error correction
3. Metagenomic classification
4. Sequence clustering
5. Sequence search
6. Abundance study

etc.

Downstream applications

- The ***size*** of the raw sequencing data makes the problem challenging
- *k*-mer counts follow ***highly skewed distributions*** making the problem computationally intensive

Application 1: Squeakr [Pandey et al. Bioinformatics '17]

Index space

Dataset	**KMC2** [Deorowicz, et al '14]	**Jellyfish2 (Bloom filter)** [Marçais & Kingsford '11]	**Squeakr (quotient filter)**	**Squeakr-exact (quotient filter)**
F. vesca	8.3	8.3	**4.8**	9.3
G. gallus	32.8	31.7	**13.0**	**28.8**
M. balbisiana	48.3	16.3	**11.1**	**14.2**
H. sapiens 1	71.4	61.8	**22.1**	**51.5**
H. sapiens 2	107.4	61.8	**30.8**	**60.1**

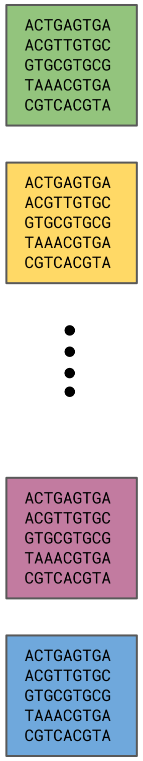
Using ***space-efficient*** data structures, we can save space and build ***simpler*** and ***efficient*** systems

Application 2: sample discovery problem

Solomon & Kingsford Nat Biotech ‘16

...ACACGTA...

Check if this new transcript has been seen before?

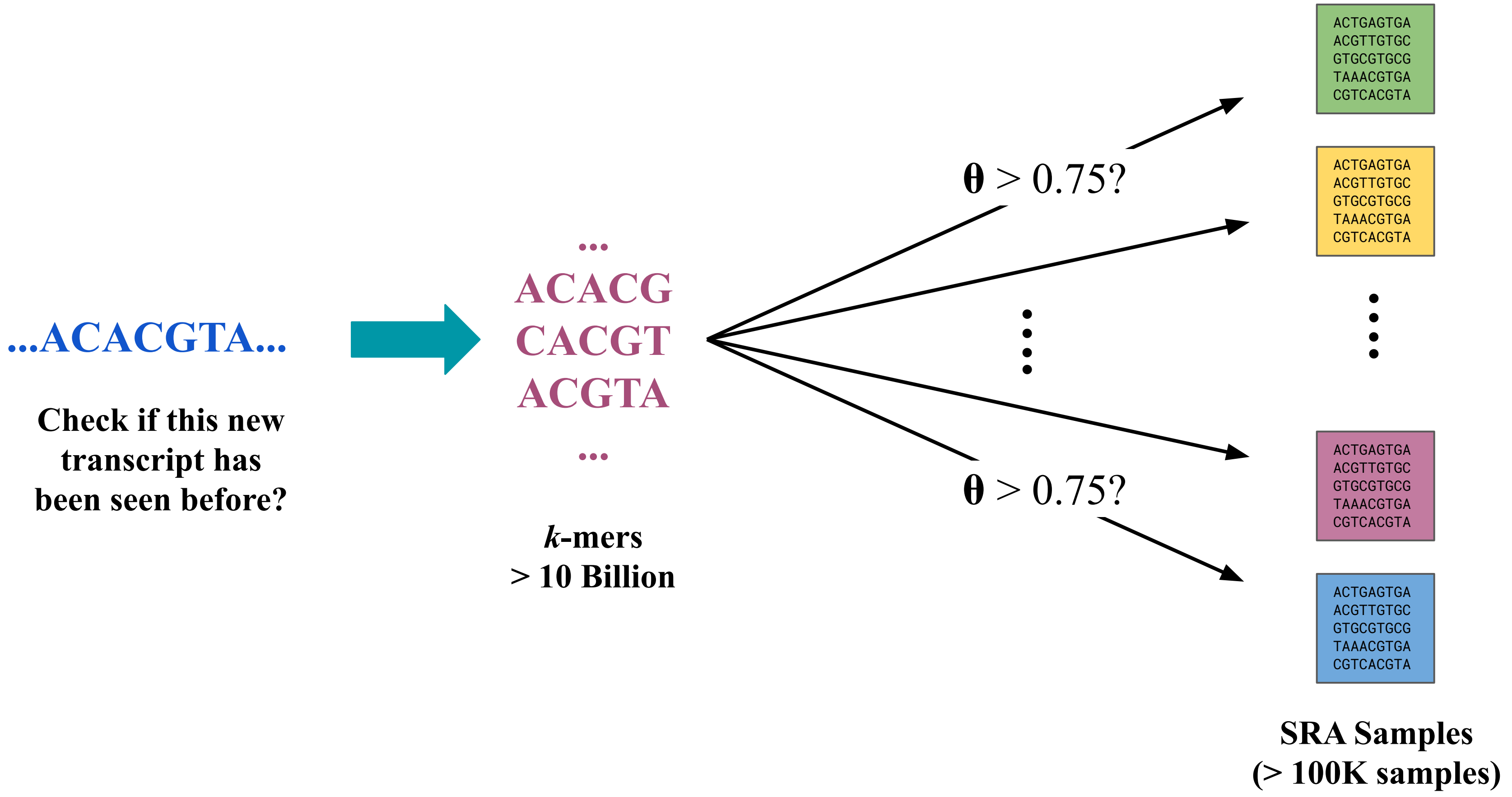


SRA Samples (> 100K samples)

Return all samples that contain at least some user-defined fraction θ of the k-mers present in the query string.

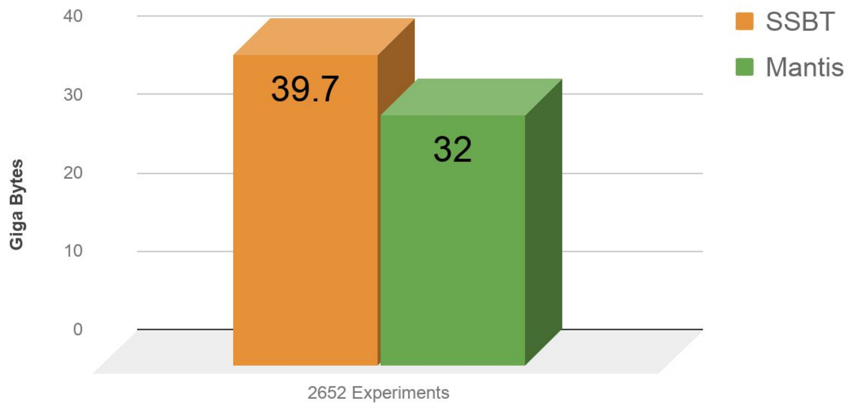
Application 2: sample discovery problem

Solomon & Kingsford Nat Biotech '16

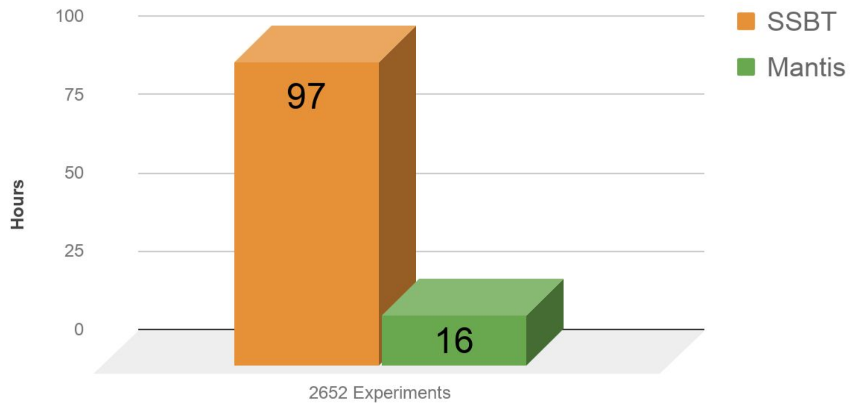


Return all samples that contain at least some user-defined fraction θ of the k-mers present in the query string

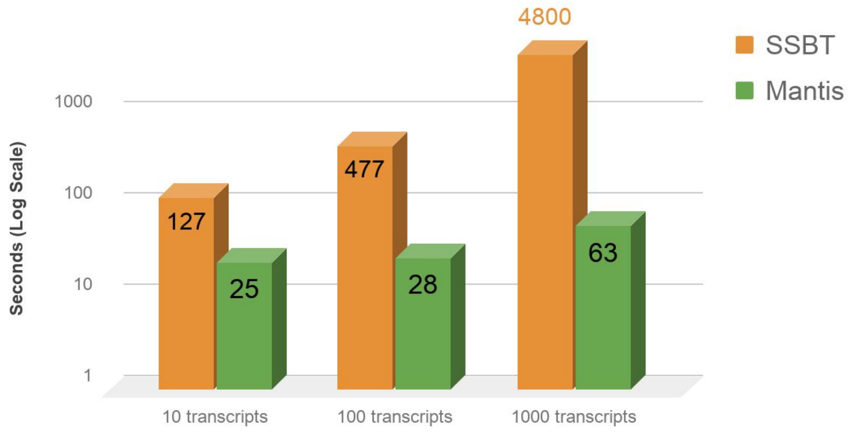
Application 2: Mantis [Pandey et al. RECOMB '18, Cell Systems '18]



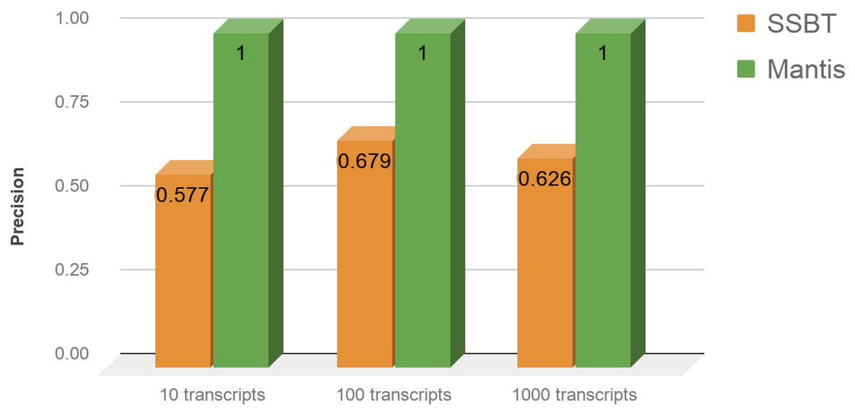
Space



Construction time



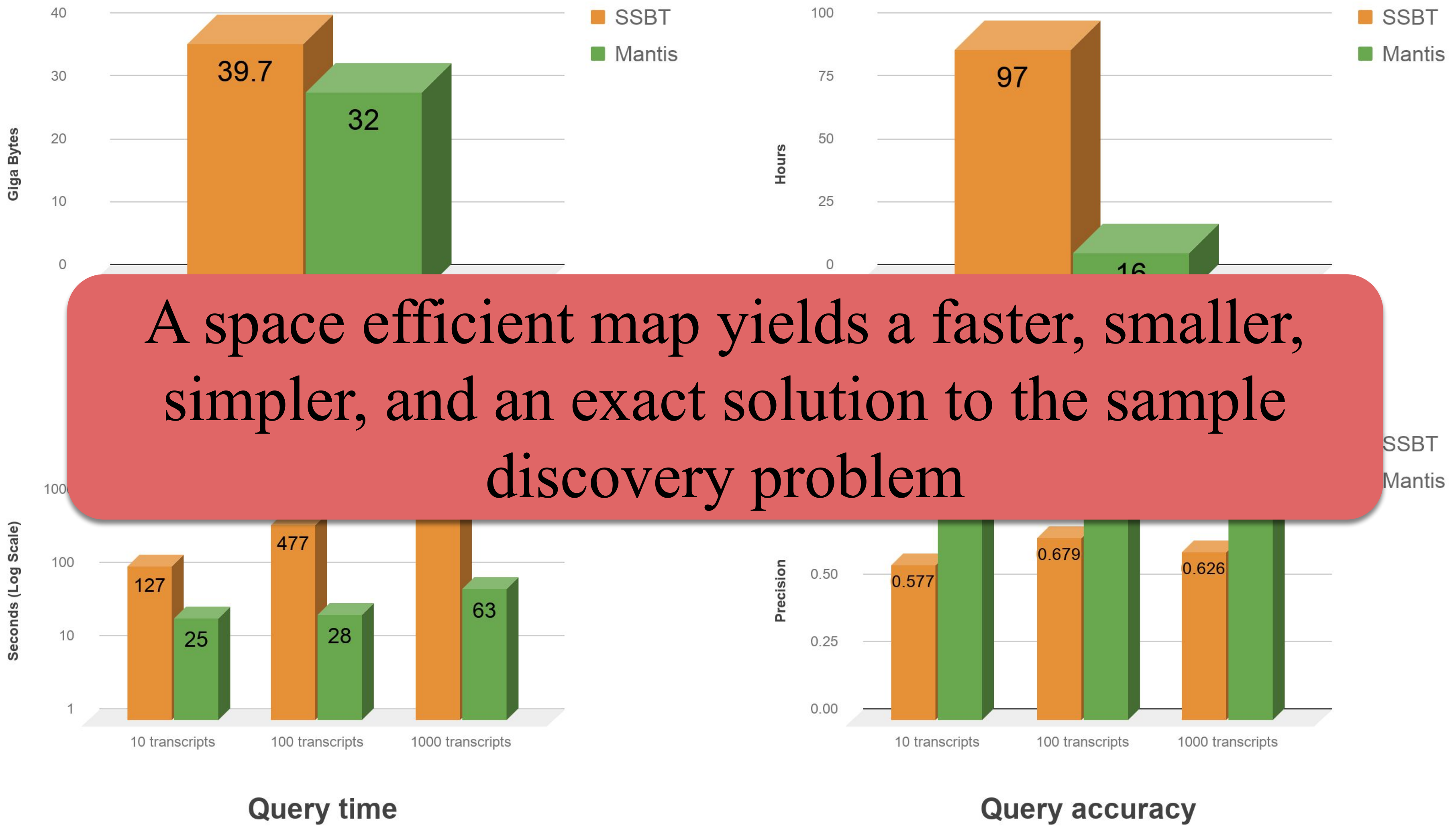
Query time



Query accuracy

SSBT (Bloom filter) [Solomon & Kingsford '17]
Mantis (quotient filter) [Pandey et al. '18]

Application 2: Mantis [Pandey et al. RECOMB '18, Cell Systems '18]

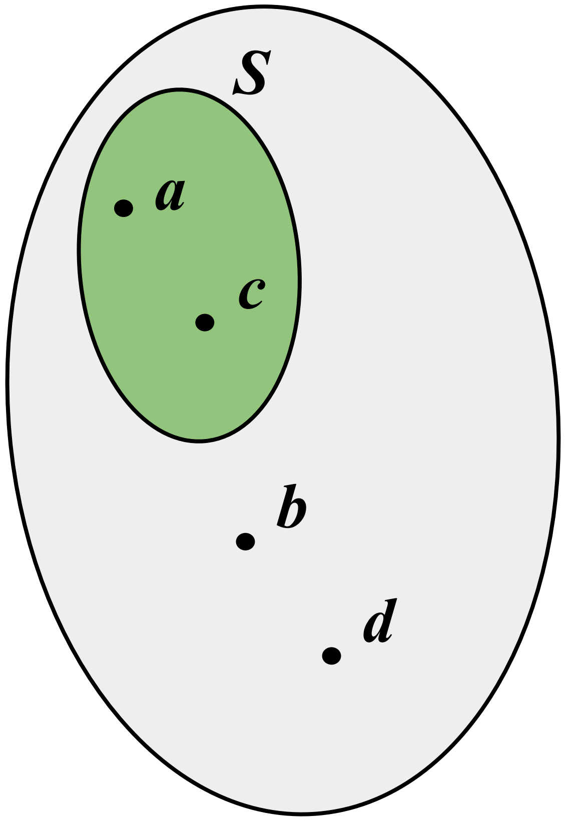


A space efficient map yields a faster, smaller, simpler, and an exact solution to the sample discovery problem

SSBT (Bloom filter) [Solomon & Kingsford '17]
Mantis (quotient filter) [Pandey et al. '18]

Dictionary data structure

A dictionary maintains a set S from universe U.



membership(a): ✓

membership(b): ✗

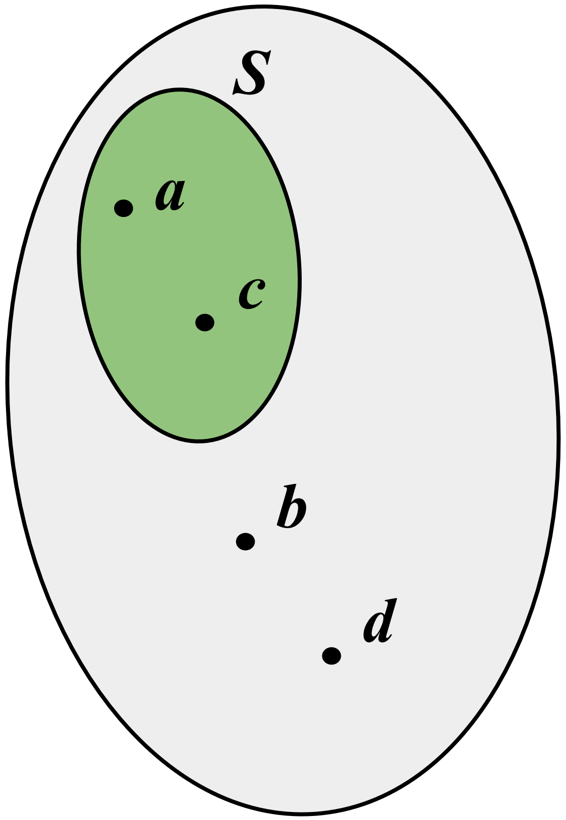
membership(c): ✓

membership(d): ✗

A dictionary supports membership queries on S.

Filter data structure

A filter is an ***approximate*** dictionary.



membership(a): ✓

membership(b): ✗

membership(c): ✓

membership(d): ✓ 👎 **false positive**

A filter supports ***<u>approximate</u>*** membership queries on S.

A filter guarantees a false-positive rate ε

if $q \in$ S, return ✓ with probability 1 **true positive**

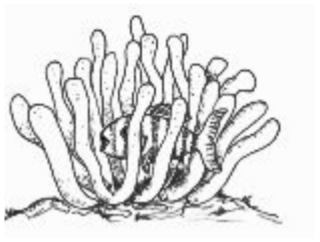
if $q \notin$ S, return
- ✗ with probability □1 - ε **true negative**
- ✓ with probability $\leq \varepsilon$ **false positive**

one-sided errors

False-positive rate enables filters to be compact

$\text{space} \geq n \log(1/\epsilon)$

$\text{space} = \Omega(n \log |U|)$



Filter

Dictionary

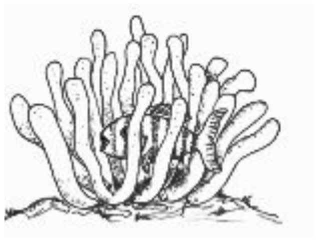
False-positive rate enables filters to be compact

$\text{space} \geq n \log(1/\epsilon)$

Small

$\text{space} = \Omega(n \log |U|)$

Large



Filter

Dictionary

For most practical purposes:

ε = 2%, a Bloom filter requires ≈ 8 bits/item

Classic filter: The Bloom filter [Bloom '70]

Bloom filters are ubiquitous (> 4300 citations)

Streaming applications

Networking

Databases

Computational biology

Storage systems

Bloom filters have suboptimal performance

	Bloom filter	Optimal
Space (bits)	$\approx 1.44\, n \log(1/\epsilon)$	$\approx n\ \log(1/\epsilon) + \Omega(n)$
CPU cost	$\Omega(1/\epsilon)$	$O(1)$
Data locality	$\Omega(1/\epsilon)$ probes	$O(1)$ probes

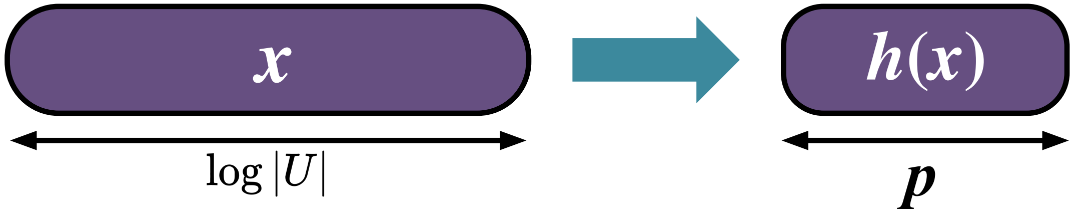
Applications often work around Bloom filter limitations

Limitations	Workarounds
No deletes	Rebuild
No resizes	Guess *N*, and rebuild if wrong
No filter merging or enumeration	???
No values associated with keys	Combine with another data structure

Bloom filter limitations increase system complexity, waste space, and slow down application performance

Quotienting is an alternative to Bloom filters
[Knuth. Searching and Sorting Vol. 3, '97]

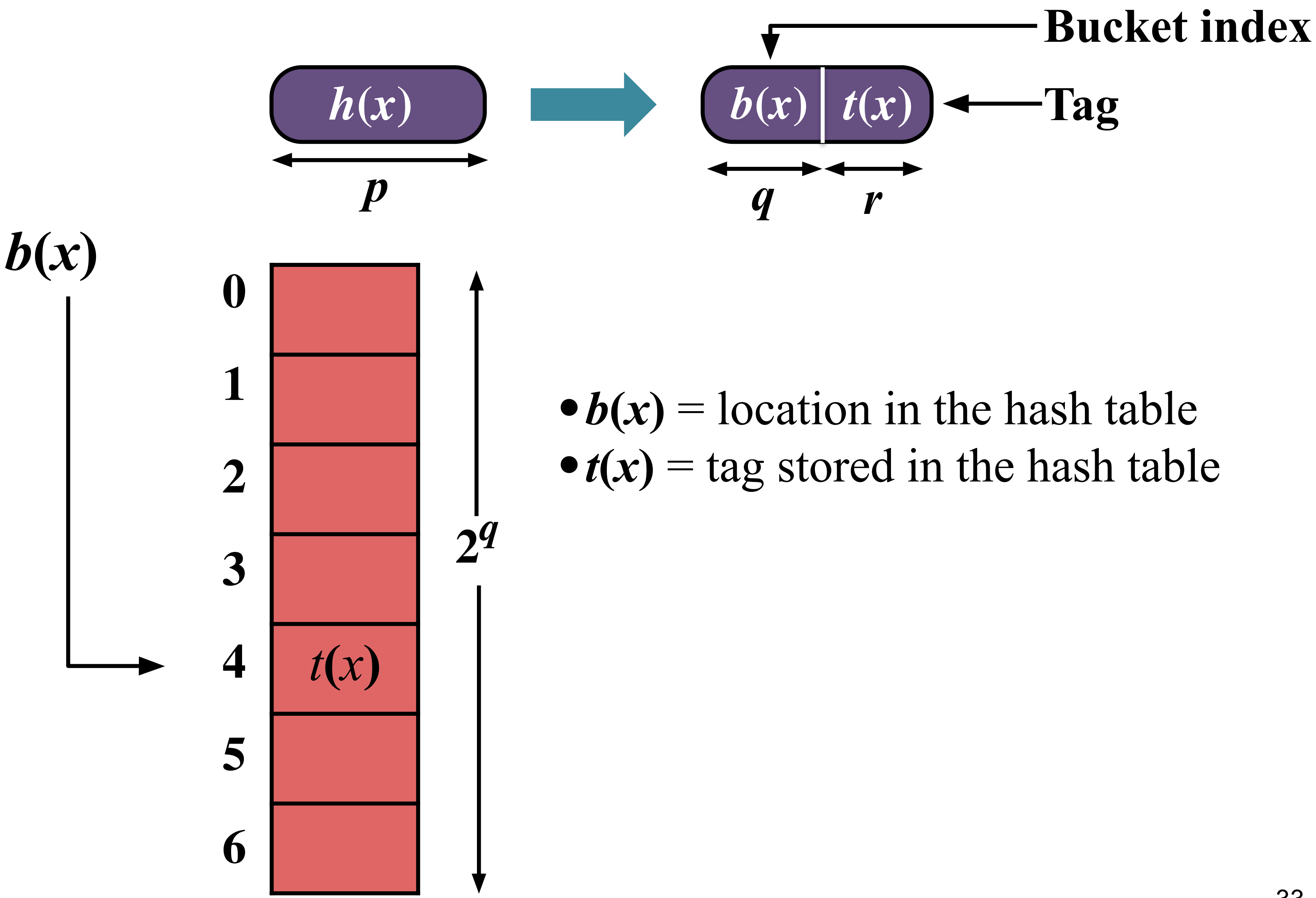
- **Store fingerprints compactly in a hash table.**
  - Take a fingerprint $h(x)$ for each element x.



- **Only source of false positives:**
  - Two distinct elements x and y, where $h(x) = h(y)$
  - If x is stored and y isn't, query(y) gives a false positives

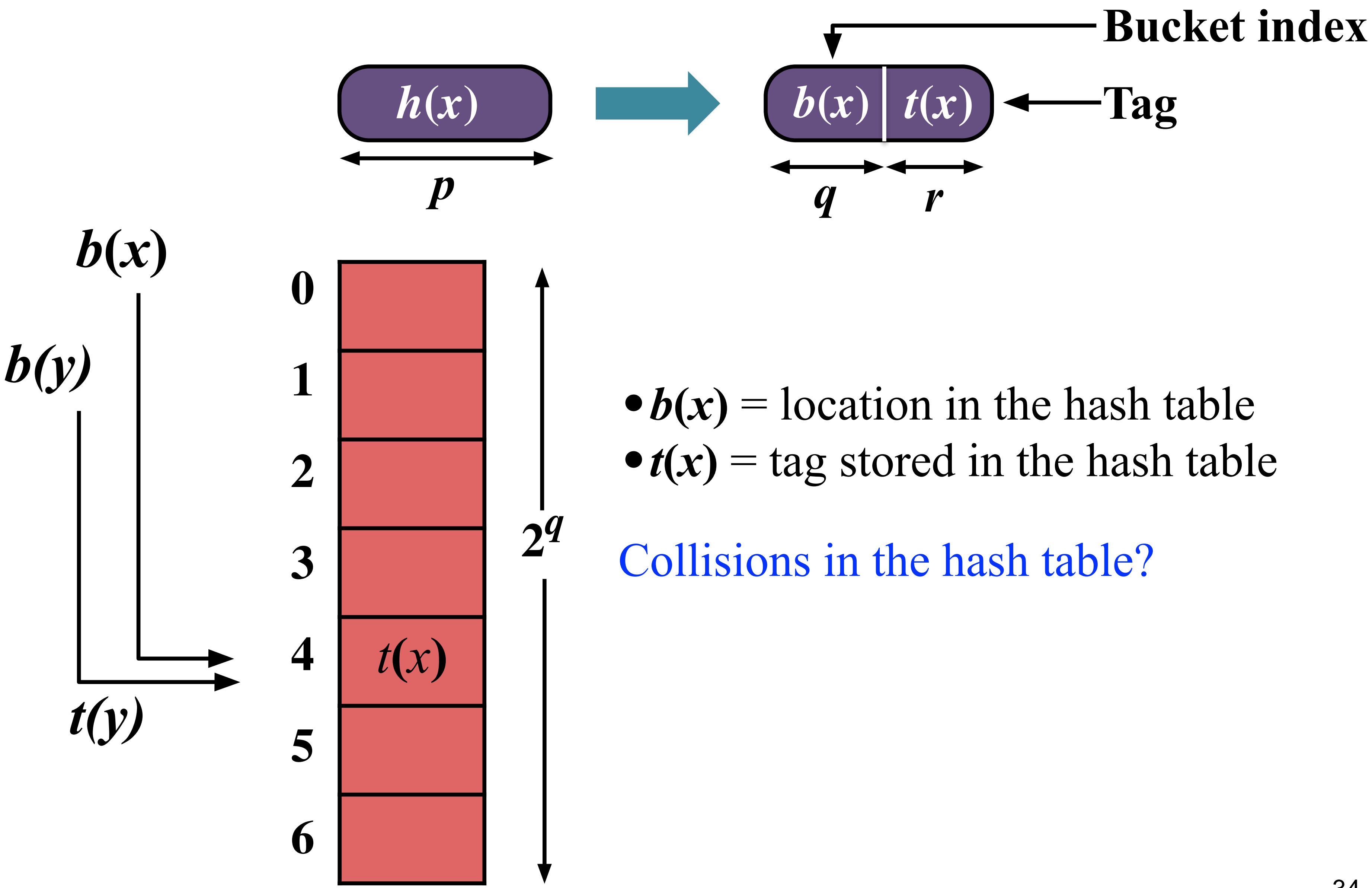
$$\Pr[\,x \text{ and } y \text{ collide}] = \frac{1}{2^p}$$

Storing fingerprints compactly



- $b(x)$ = location in the hash table
- $t(x)$ = tag stored in the hash table

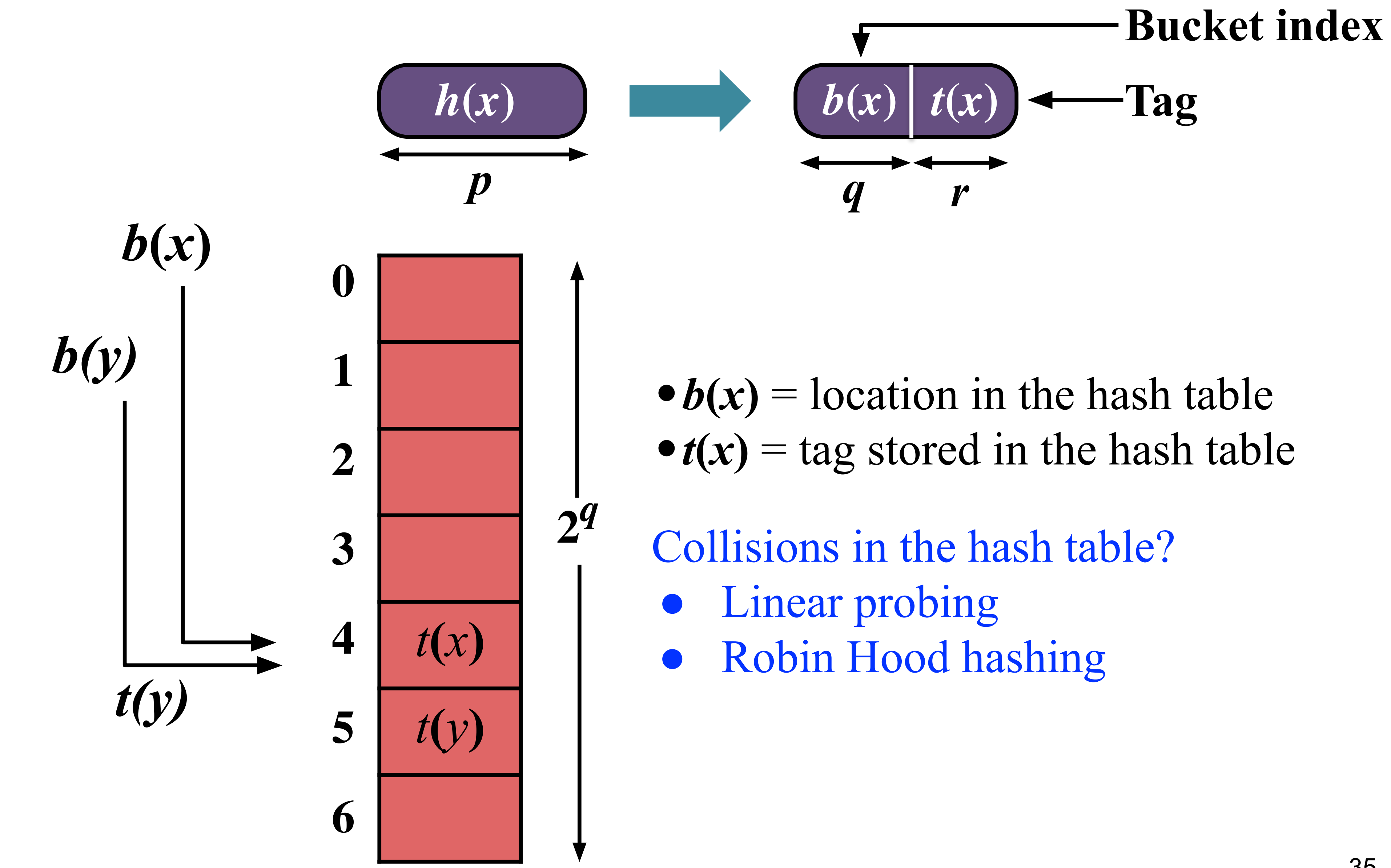
Storing fingerprints compactly



- $b(x)$ = location in the hash table
- $t(x)$ = tag stored in the hash table

Collisions in the hash table?

Storing fingerprints compactly

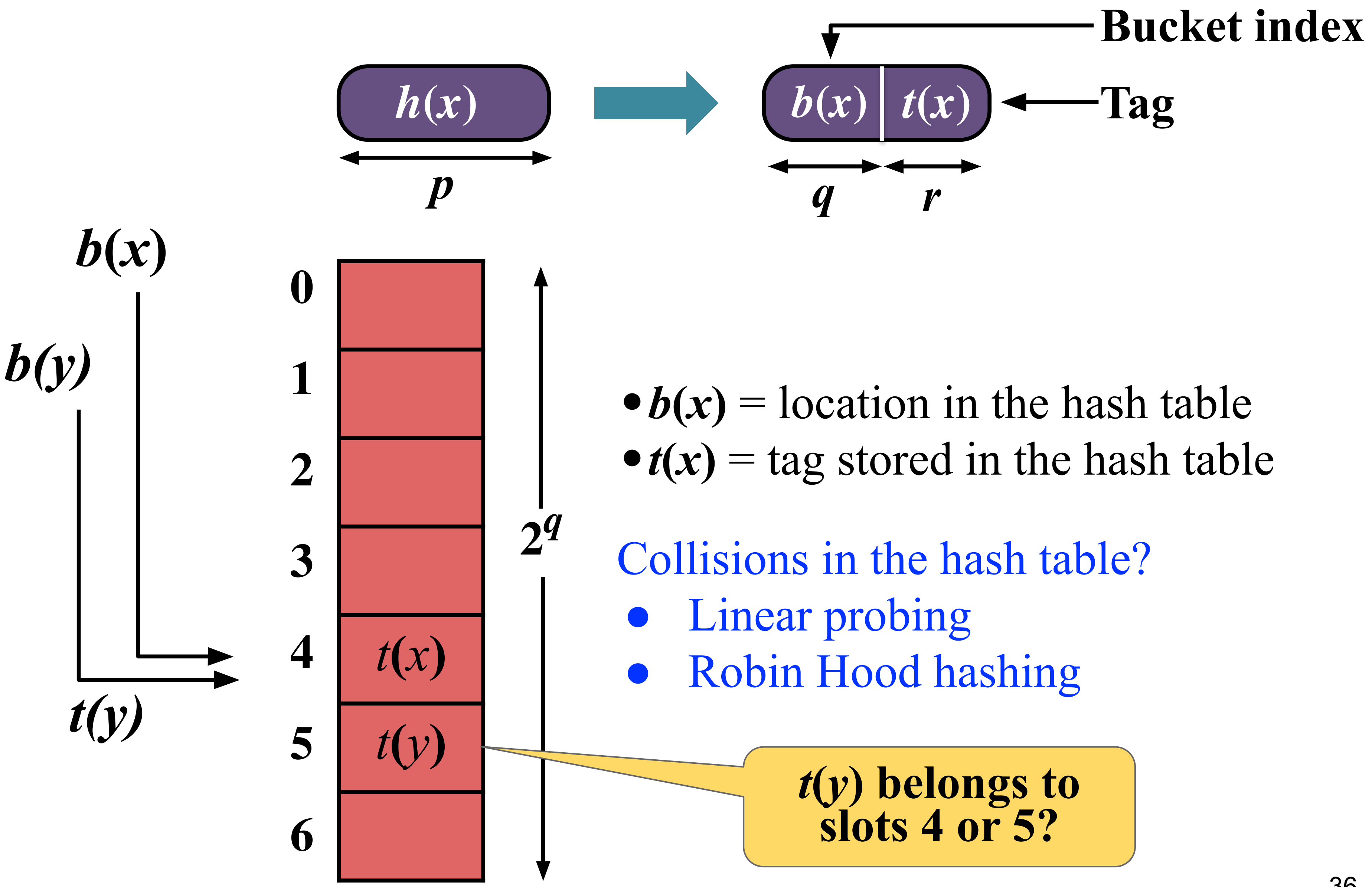


- ***b*(*x*)** = location in the hash table
- ***t*(*x*)** = tag stored in the hash table

Collisions in the hash table?

- Linear probing
- Robin Hood hashing

Storing fingerprints compactly



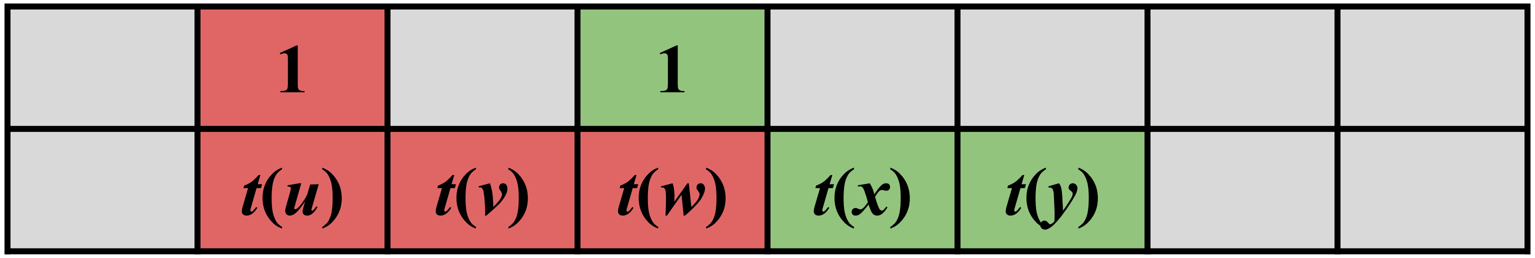
- $b(x)$ = location in the hash table
- $t(x)$ = tag stored in the hash table

Collisions in the hash table?

- Linear probing
- Robin Hood hashing

Resolving collisions in the QF [Pandey et al. SIGMOD ‘17]

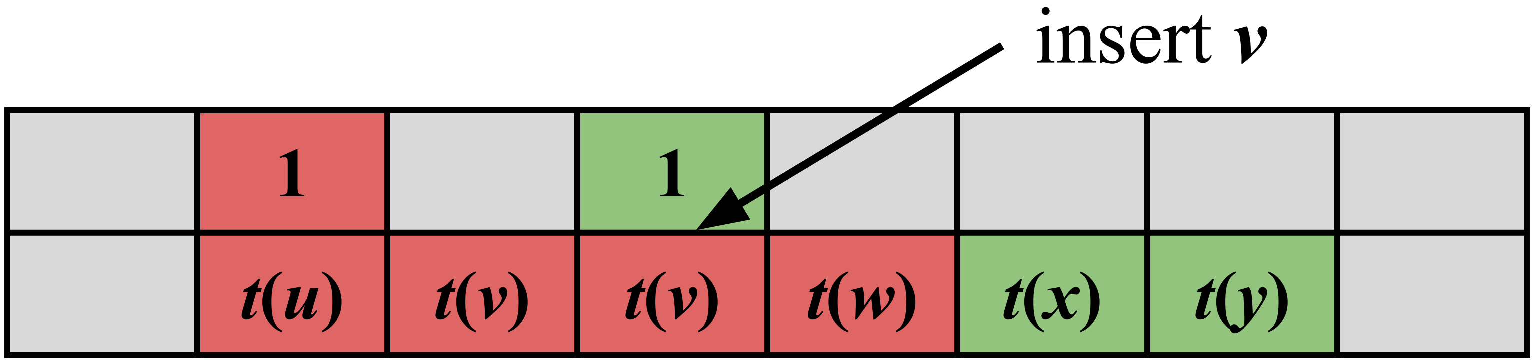
- QF uses two metadata bits to resolve collisions and identify home bucket



- The metadata bits group tags by their home bucket

Resolving collisions in the QF [Pandey et al. SIGMOD '17]

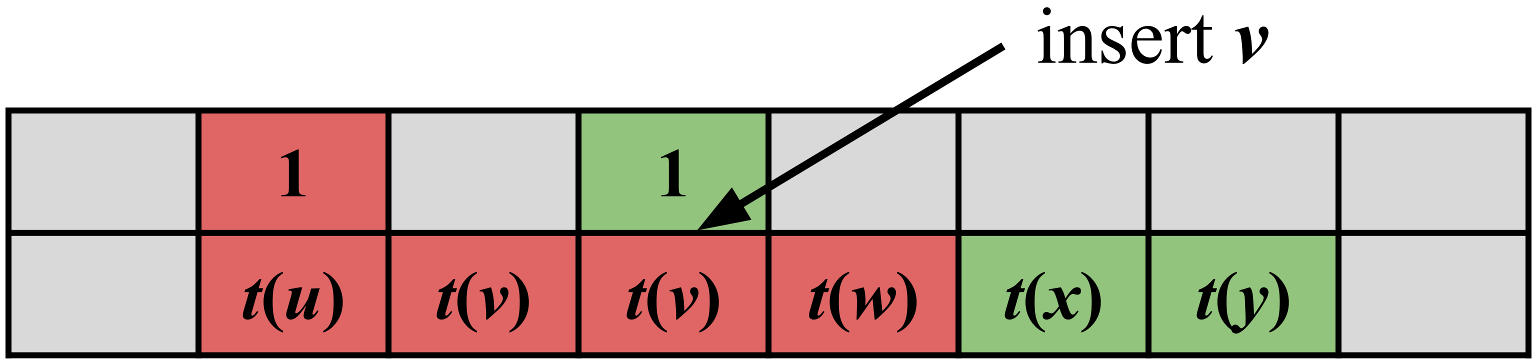
- QF uses two metadata bits to resolve collisions and identify home bucket



- The metadata bits group tags by their home bucket

Resolving collisions in the QF [Pandey et al. SIGMOD '17]

- QF uses two metadata bits to resolve collisions and identify home bucket



- The metadata bits group tags by their home bucket

The metadata bits enable us to identify the slots holding the contents of each bucket.

More

Quotienting enables many features in the QF

- Good cache locality
- Efficient scaling out-of-RAM
- Deletions
- Enumerability/Mergeability
- Resizing
- Maintains count estimates or associate values
- Uses variable-sized encoding for counts **[Counting quotient filter]**
    - **Asymptotically optimal space: $O(\sum |C(x)|)$**

Quotienting enables many features in the QF

- Good cache locality
- Efficient scaling out-of-RAM
- Deletions
- Enumerability/Mergeability
- Resizing
- Maintains count estimates or associate values
- Uses variable-sized encoding for counts **[Counting quotient filter]**
    - **Asymptotically optimal space: $O(\sum |C(x)|)$**

Mantis uses the QF to map small keys to values

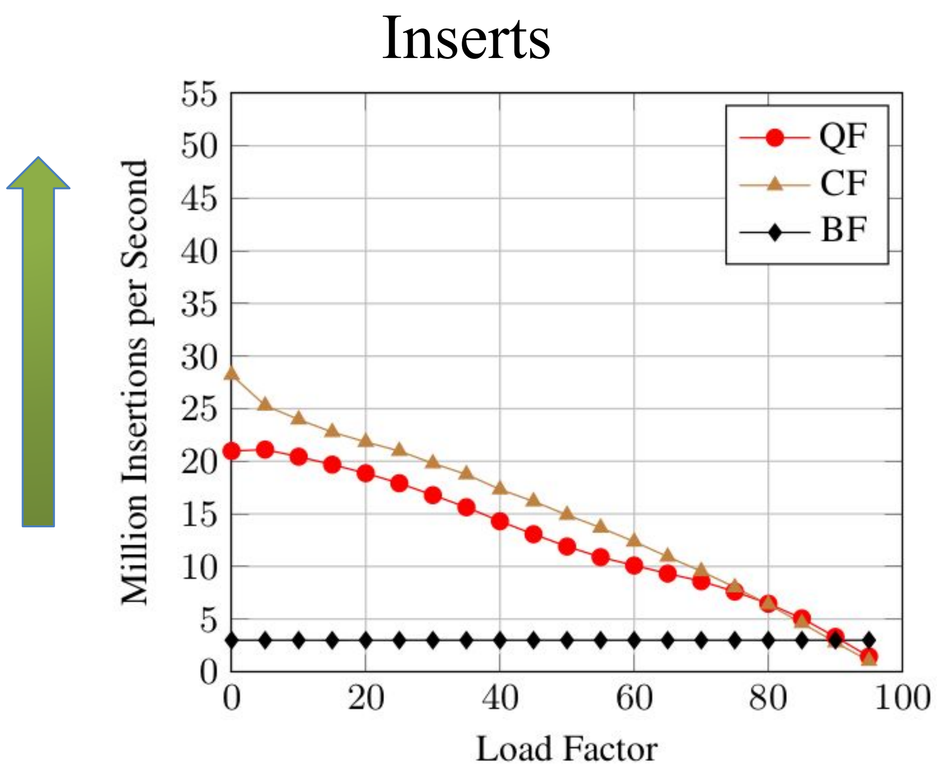
Squeakr uses the QF to count items

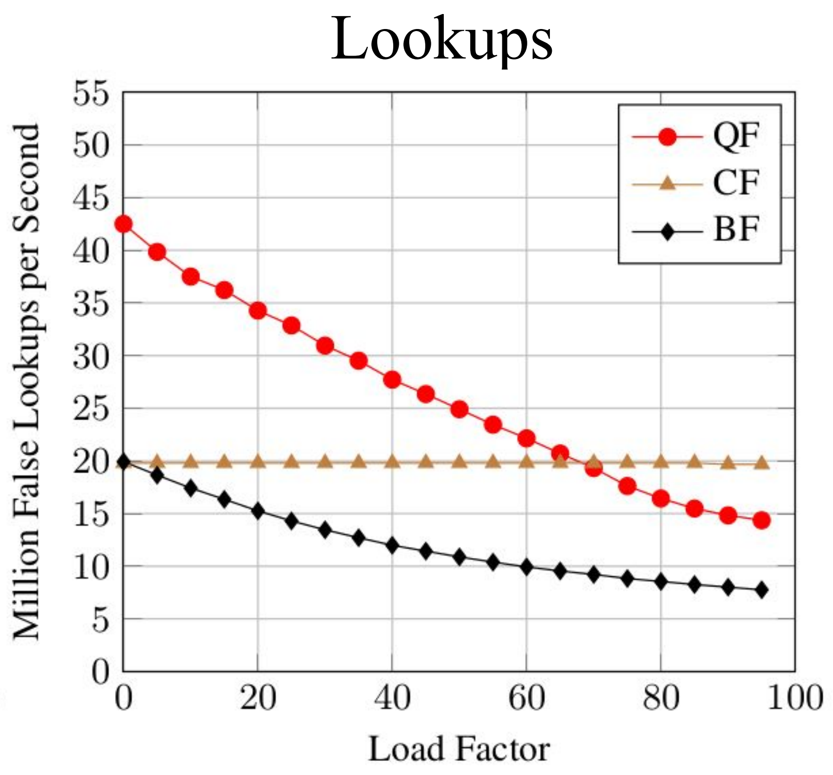
Quotient filters use less space than Bloom filters for all practical configurations

	Quotient filter	Bloom filter	Optimal
Space (bits)	$\approx n \log(1/\epsilon) + 2.125n$	$\approx 1.44\, n \log(1/\epsilon)$	$\approx n \log(1/\epsilon) + \Omega(n)$
CPU cost	$O(1)$ expected	$\Omega(1/\epsilon)$	$O(1)$
Data locality	1 probe + scan	$\Omega(1/\epsilon)$ probes	$O(1)$ probes

The quotient filter has theoretical advantages over the Bloom filter

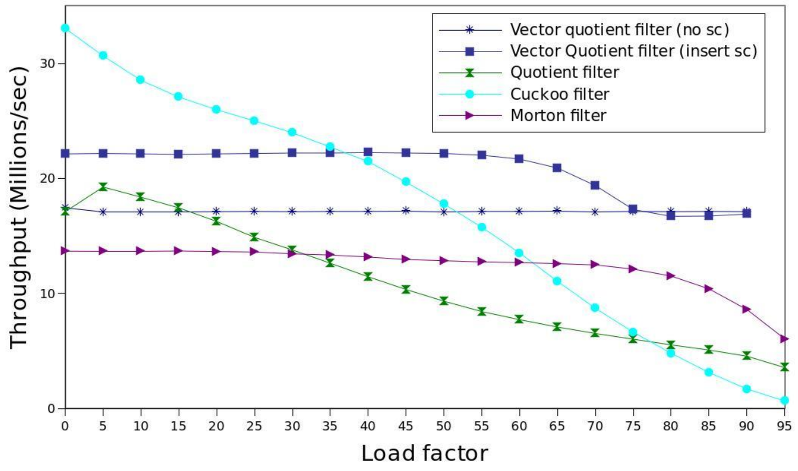
Quotient filters perform better (or similar) to other non-counting filters

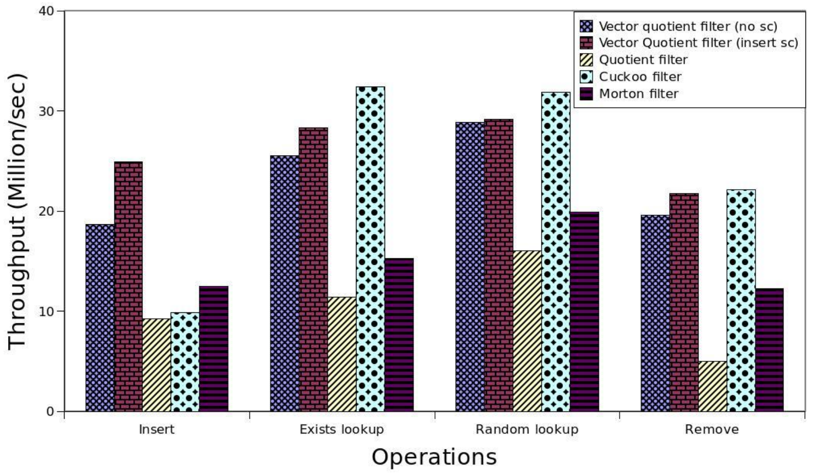




- Insert performance is similar to the state-of-the-art non-counting filters
- Query performance is significantly fast at low load-factors and slightly slower at higher load-factors

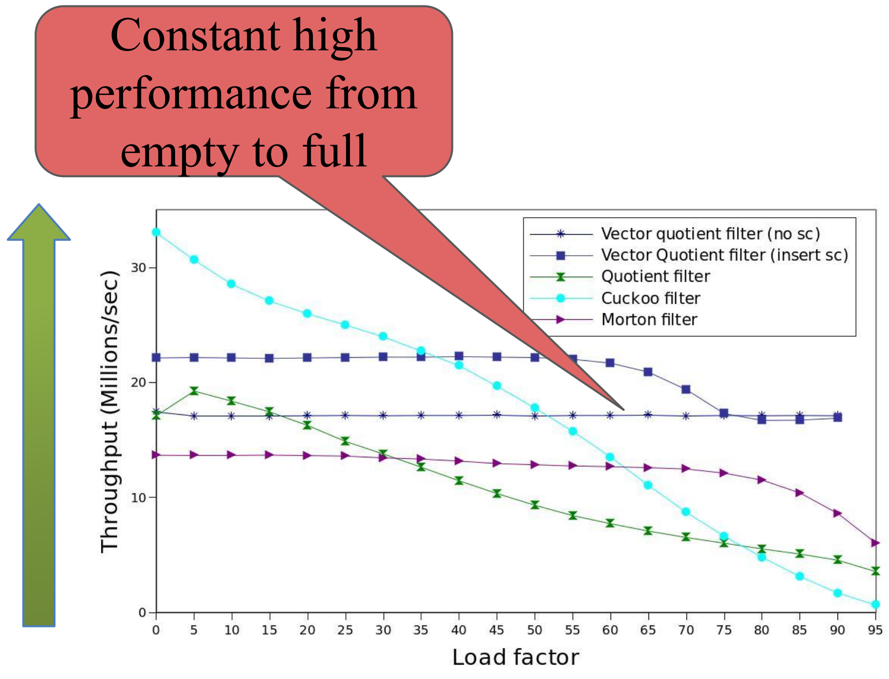
Vector quotient filter (VQF)[Pandey et al. SIGMOD '21]

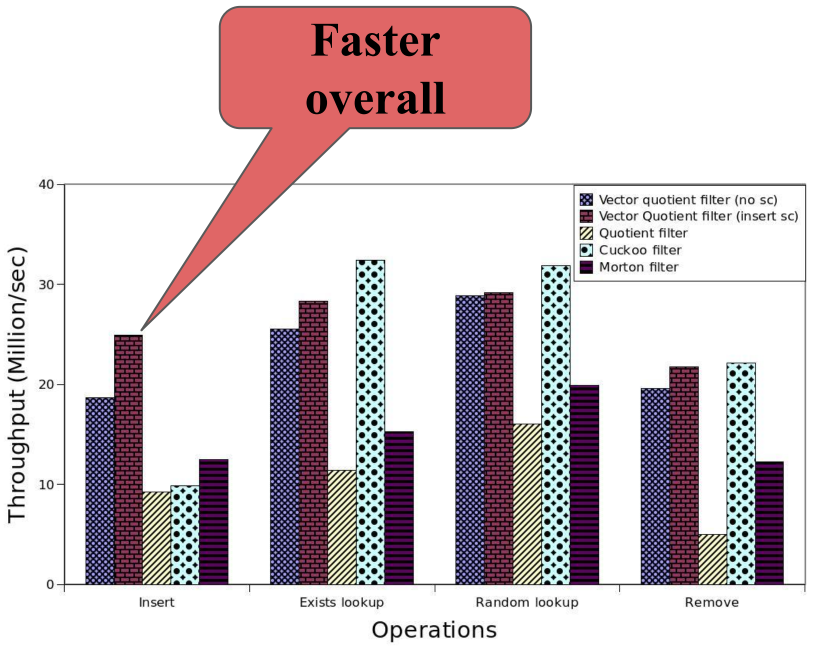

Throughput (Millions/sec)
Load factor
Vector quotient filter (no sc)
Vector Quotient filter (insert sc)
Quotient filter
Cuckoo filter
Morton filter



Combining hashing techniques (**Robin Hood + 2-choice hashing**)
Using ultra-wide vector operations (**AVX512-BW**)

Vector quotient filter (VQF)[Pandey et al. SIGMOD '21]





Combining hashing techniques (**Robin Hood + 2-choice hashing**)
Using ultra-wide vector operations (**AVX512-BW**)

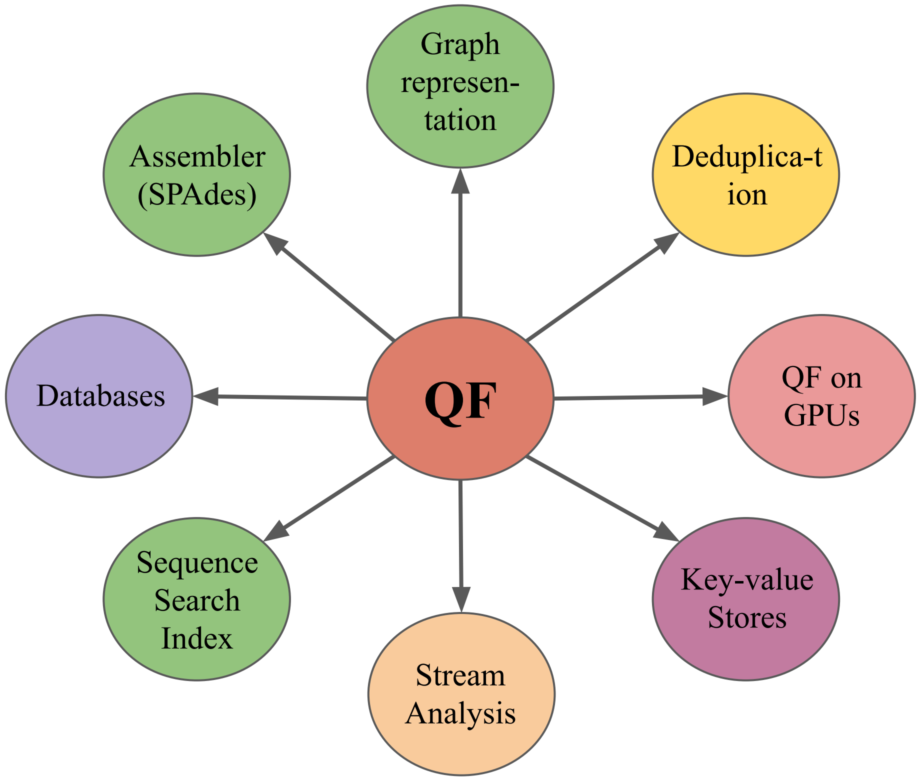
Quotient filter's impact in computer science

Computational biology

1. Squeakr
2. deBGR
3. Mantis
4. VariantStore

Databases/Systems

1. Anomaly detection
2. BetrFS file system



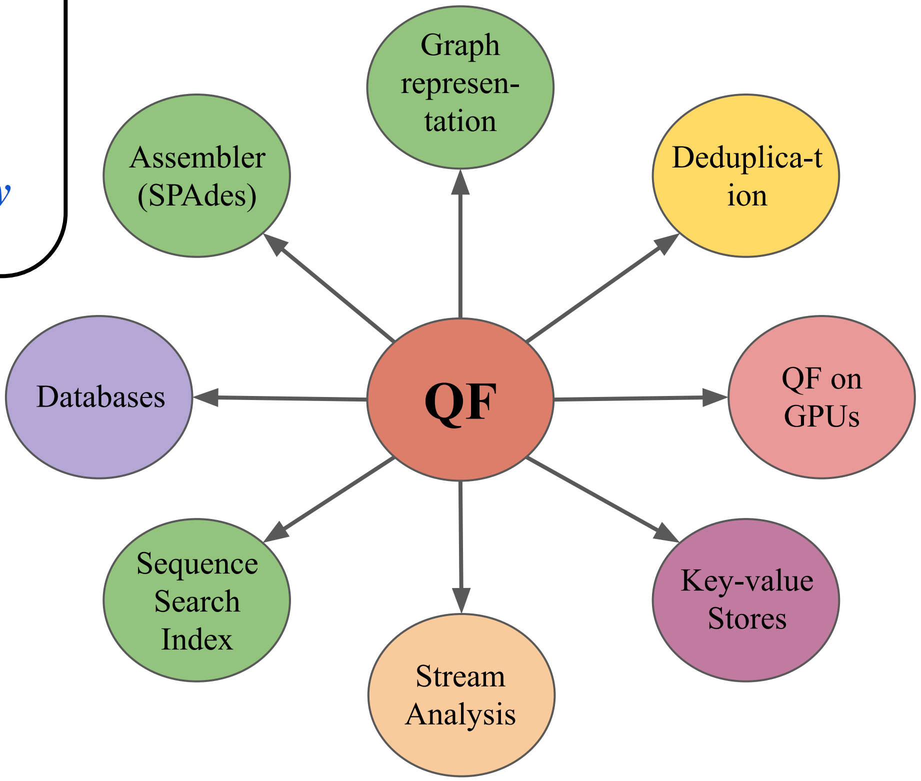
Theoretically well-founded data structures can have a ***big impact*** on multiple subfields across ***academia and industry***

Quotient filter's impact in computer science

Computational biology
1. Squeakr
2. deBGR
3. Mantis
4. VariantStore
5. SPAdes assembler
6. Khmer software
7. MQF

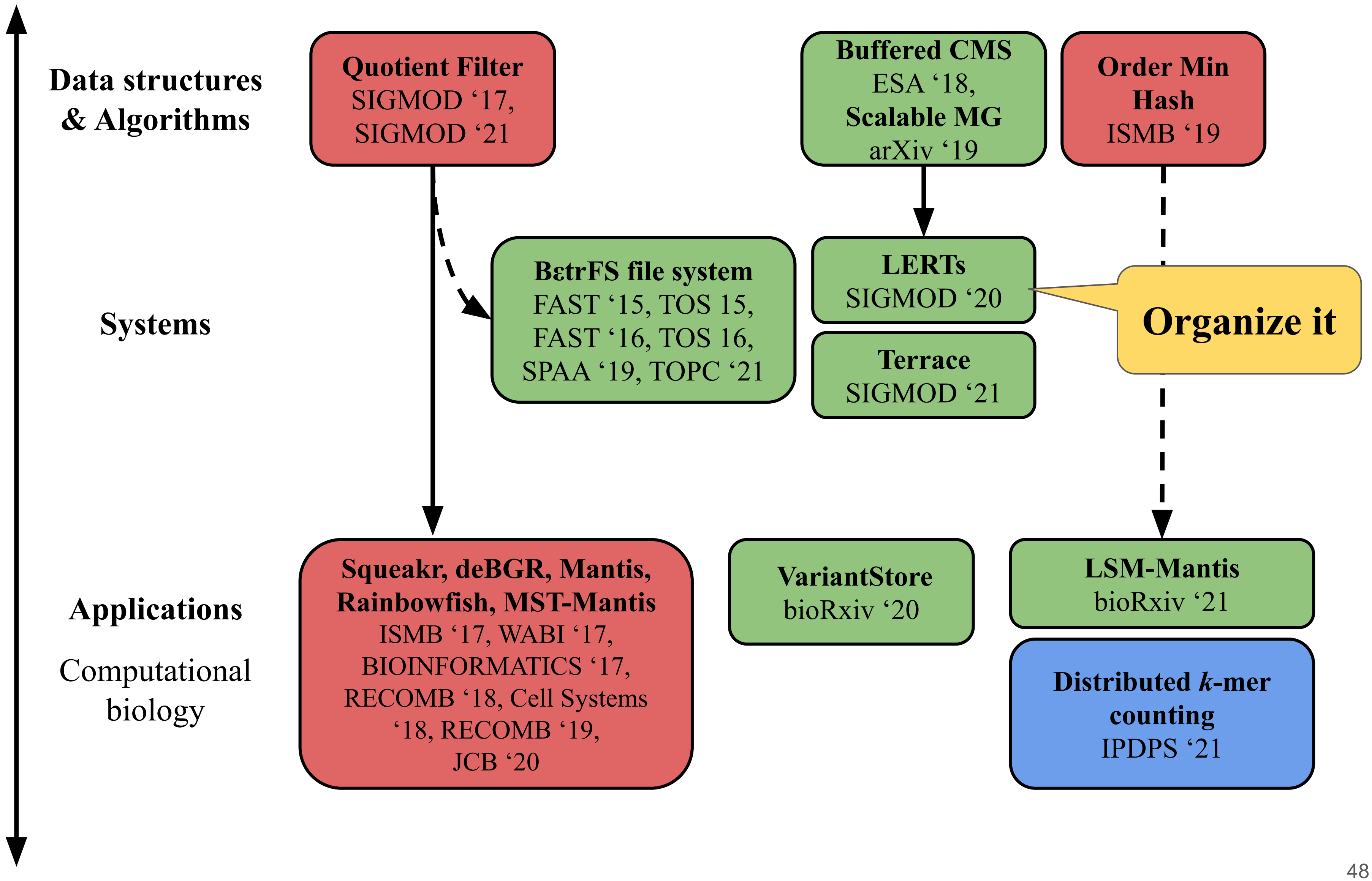
Industry
1. VMware
2. Nutanix
3. Apocrypha
4. Hyrise
5. ***A data security startup***

Databases/Systems
1. Anomaly detection
2. BetrFS file system
3. Counting on GPUs
4. Concurrent filters



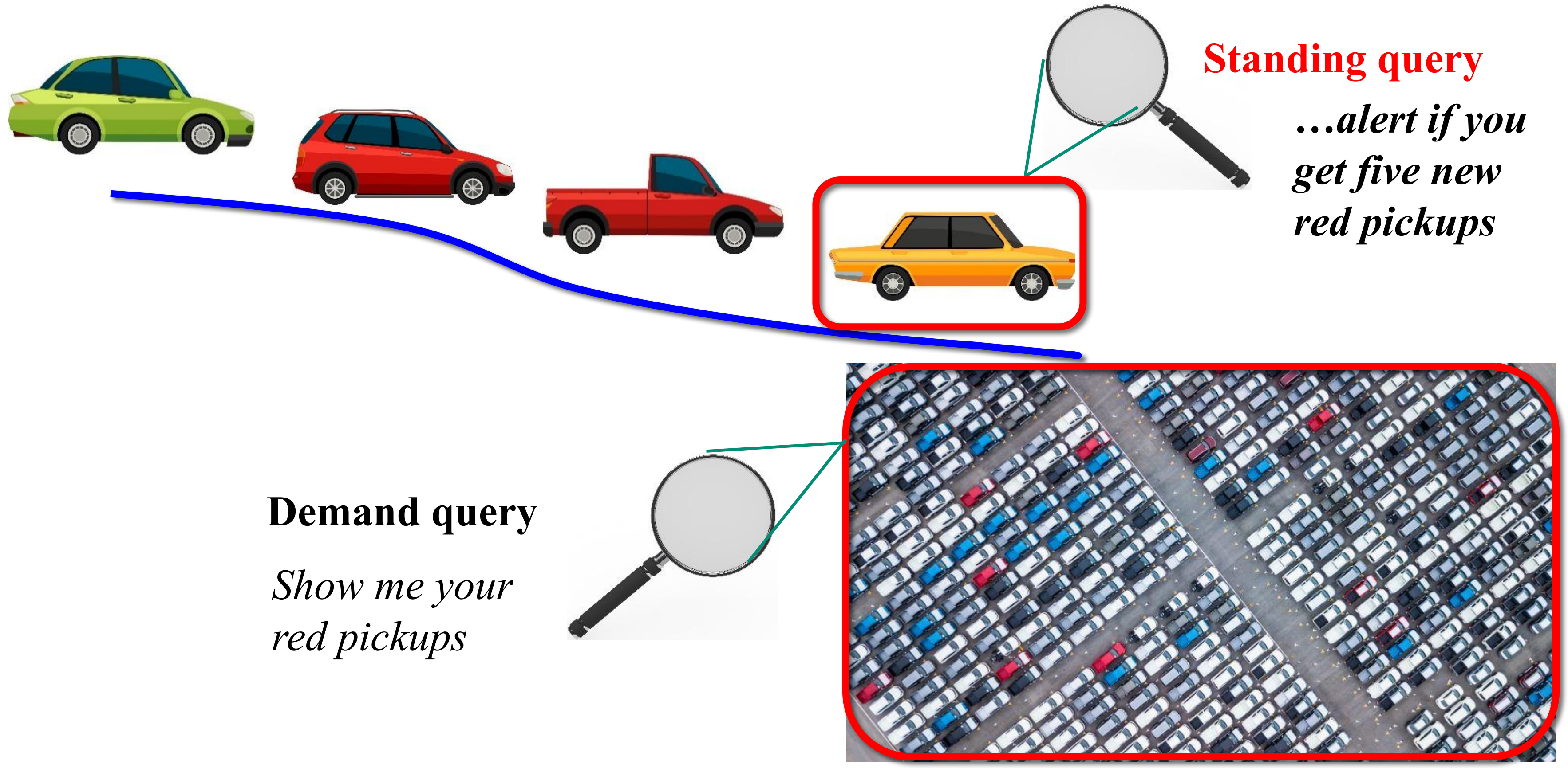
Theoretically well-founded data structures can have a ***big impact*** on multiple subfields across ***academia and industry***

Learned “Shrink it”. Now “Organize it”



Data structures & Algorithms
Systems
Applications
Computational biology
Quotient Filter SIGMOD ‘17, SIGMOD ‘21
Buffered CMS ESA ‘18, Scalable MG arXiv ‘19
Order Min Hash ISMB ‘19
BεtrFS file system FAST ‘15, TOS 15, FAST ‘16, TOS 16, SPAA ‘19, TOPC ‘21
LERTs SIGMOD ‘20
Terrace SIGMOD ‘21
Organize it
Squeakr, deBGR, Mantis, Rainbowfish, MST-Mantis ISMB ‘17, WABI ‘17, BIOINFORMATICS ‘17, RECOMB ‘18, Cell Systems ‘18, RECOMB ‘19, JCB ‘20
VariantStore bioRxiv ‘20
LSM-Mantis bioRxiv ‘21
Distributed k-mer counting IPDPS ‘21

Timely event detection problem


Standing query
…alert if you get five new red pickups
Demand query
Show me your red pickups

In **timely event-detection problem (TED)**, we want to answer standing queries

https://firehose.sandia.gov

Features we need in the solution

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion

Features we need in the solution

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion

- Events are high-consequence real-life events

No false-negatives; few false-positives

Sampling

Timely reporting (real-time)

Features we need in the solution

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion

- Events are high-consequence real-life events

No false-negatives; few false-positives

Timely reporting (real-time)

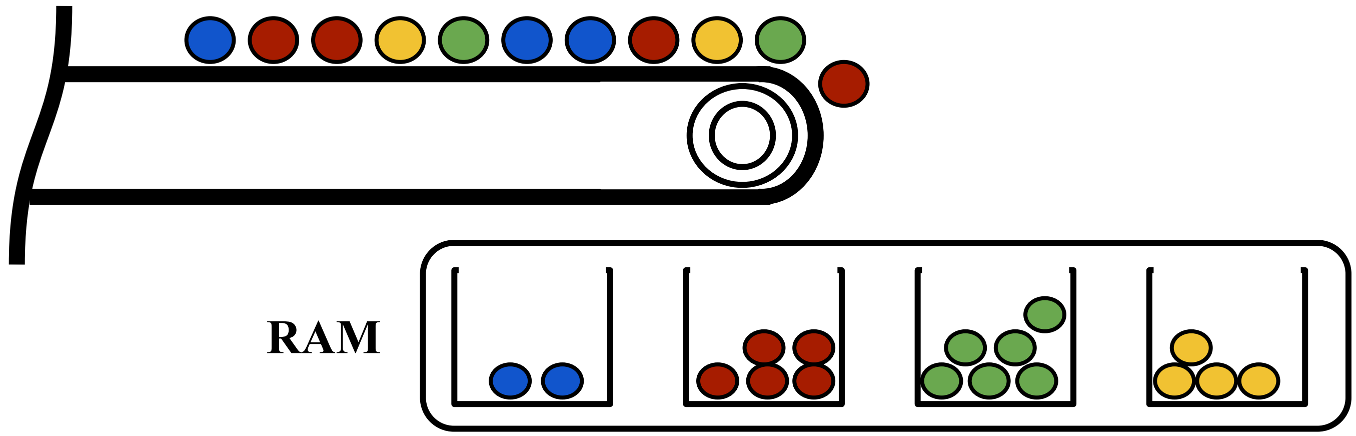
Sampling

- Malicious traffic forms a small portion of the stream

Very small reporting thresholds

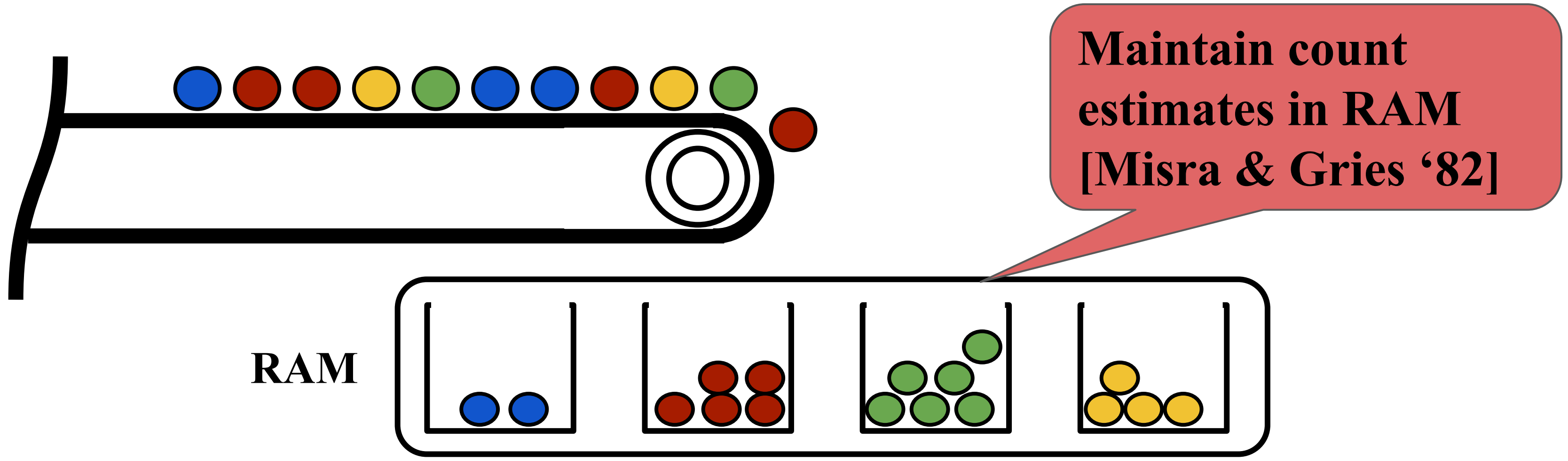
One-pass streaming has errors

- **Heavy hitter problem:** report items whose frequency $\geq \varphi N$
- Exact one-pass solution solution requires $\Omega(N)$ space



One-pass streaming has errors

- **Approximate solution**: report all items with count ≥ ***φN*,** none with < ***(φ−ε)N*** [Alon et al. 96, Berinde et al. 10, Bhattacharyya et al. 16, Bose et al. 03, Braverman et al. 16, Charikar et al. 02, Cormode et al. 05, Demaine et al. 02, Dimitropoulos et al. 08, Larsen et al. 16, Manku et al. 02.]
- Approximate solutions require **Ω(1/*ε*)** space

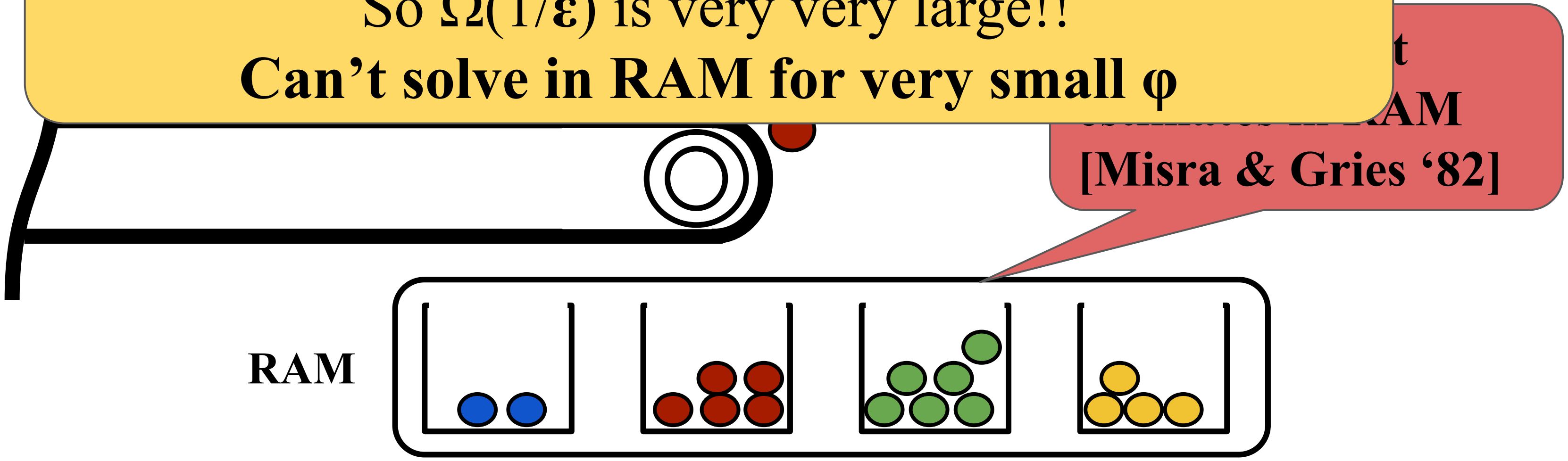


Real time with false-positives!

One-pass streaming has errors

- **Approximate solution**: report all items with count ≥ ***φN*,** none with < ***(φ−ε)N*** [Alon et al. 96, Berinde et al. 10, Bhattacharyya et al. 16, Bose et al. 03, Braverman et al. 16, Charikar et al. 02, Cormode et al. 05, Demaine et al. 02, Dimitropoulos et al. 08, Larsen et al. 16, Manku et al. 02.]
- Approximate solutions require **Ω(1/ε)** space

For some apps, φ*N* is a small constant,
So Ω(1/**ε**) is very very large!!
Can't solve in RAM for very small φ



Real time with false-positives!

One-pass solution has:

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion ✓

- Events are high-consequence real-life events

No false-negatives; few false-positives ✓

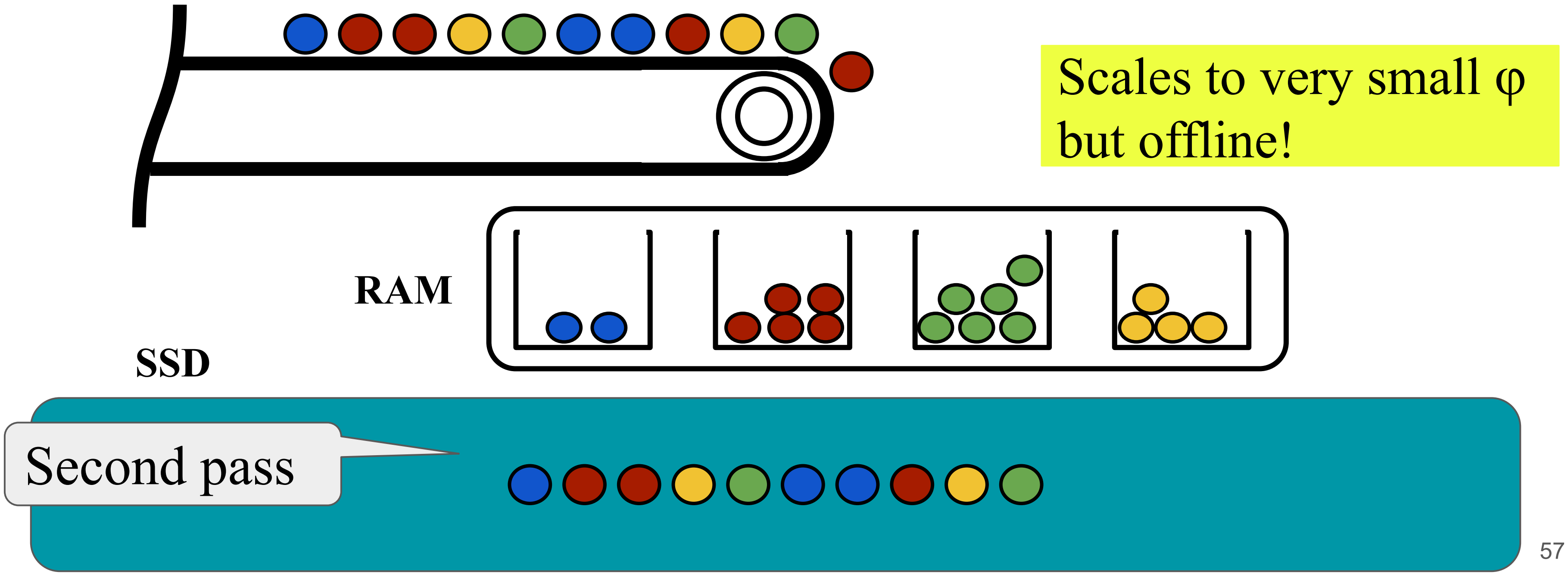
Timely reporting (real-time) ✓

- Malicious traffic forms a small portion of the stream

Very small reporting thresholds ✗

Two-pass streaming isn't real-time

- A second pass over the stream can get rid of errors
- Store the stream on SSD and access it later



Two-pass solution has:

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion

- Events are high-consequence real-life events

No false-negatives; few false-positives ✓

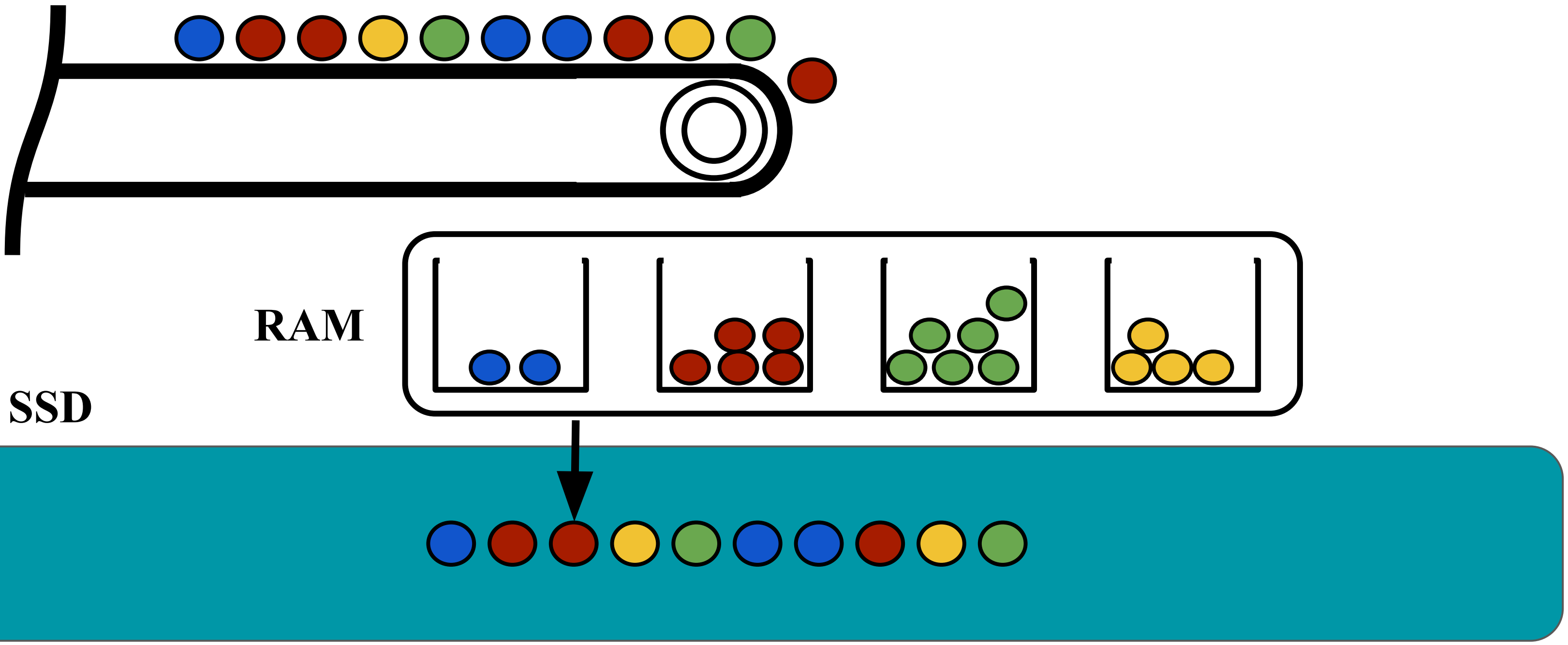
Timely reporting (real-time) ✗

- Malicious traffic forms a small portion of the stream

Very small reporting thresholds ✓

If data is stored: why not access it?

Why wait for second pass?



Idea: combine streaming and external memory (EM)[Pandey et al. SIGMOD '20]

Streaming model

External memory algorithms

Use an efficient external-memory counting data structure to scale Misra-Gries algorithm to SSDs

Operations in external memory dictionaries

Bender et al. ‘12

Insert	Query
$O\left(\frac{1}{B}\log\frac{N}{M}\right)$	$O\left(\log\frac{N}{M}\right)$

Performance bounds are parameterized by block transfer size B, memory size M, data size N.

Operations in external memory dictionaries

Bender et al. ‘12

Insert	Query
$O\left(\frac{1}{B}\log\frac{N}{M}\right)$	$O\left(\log\frac{N}{M}\right)$

< 1 I/O per observation

Performance bounds are parameterized by block transfer size B, memory size M, data size N.

Operations in external memory dictionaries

Bender et al. ‘12

Insert	Query
$O\left(\frac{1}{B}\log\frac{N}{M}\right)$	$O\left(\log\frac{N}{M}\right)$

< 1 I/O per observation

> 1 I/O per observation

Perfo re parameterized b size B, memory size M, data size N.

EM dictionary doesn’t have real-time reporting

Bender et al. ‘12

But every insert is also a query in real-time reporting!

Insert	Query
$O\left(\frac{1}{B}\log\frac{N}{M}\right)$	$O\left(\log\frac{N}{M}\right)$

<1 I/O per observation

>1 I/O per observation

Performance bounds are parameterized by block transfer size B, memory size M, data size N.

EM dictionary doesn't have real-time reporting

Bender et al. '12

But every insert is also a query in real-time reporting!

Insert	Query
$O\left(\frac{1}{B}\log\frac{N}{M}\right)$	$O\left(\log\frac{N}{M}\right)$

Traditional EM dictionary doesn't solve the problem!

observation

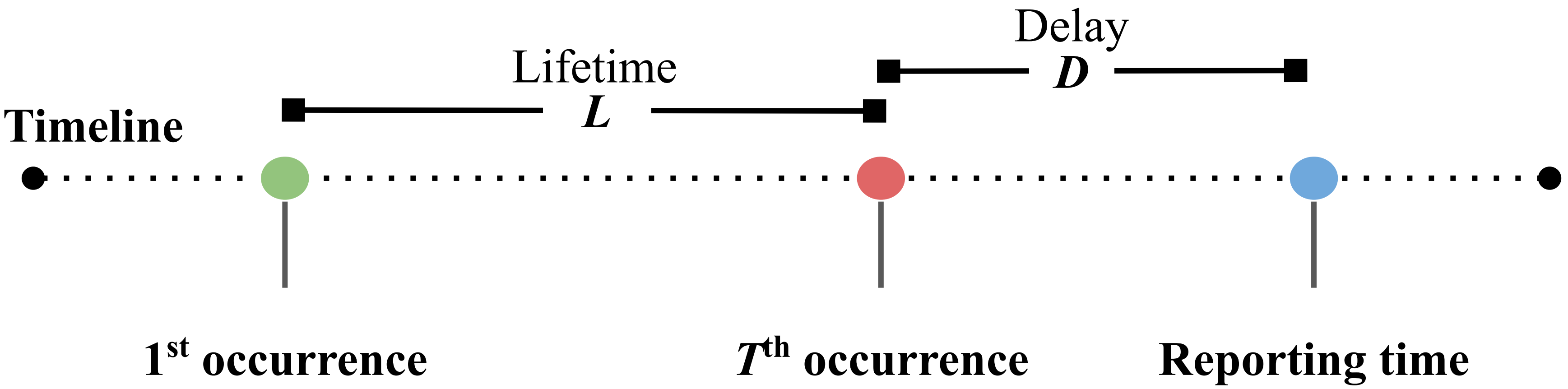
observation

Perfo re parameterized b size B, memory size M, data size N.

Idea: reporting with bounded delay

We define the time stretch of a report to be

$$\text{Time stretch} = 1 + \alpha = 1 + \frac{\text{Delay}}{\text{Lifetime}}$$



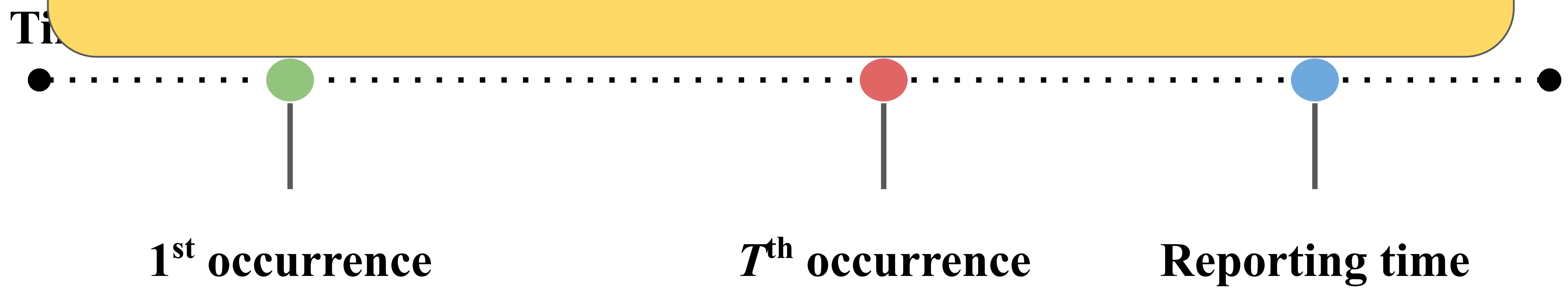
More

Idea: reporting with bounded delay

We define the time stretch of a report to be

Time stretch = 1 + α = 1 + $\frac{\text{Delay}}{}$

Main idea: the longer the lifetime of an item, the more leeway we have in reporting it



More

Leveled External-Memory Reporting Table (LERT) [Pandey et al. SIGMOD '20]

- Given a stream of size N and $\varphi N > \boldsymbol{\Omega(N/M)}$ the amortized cost of solving real-time event detection is

$$O\left(\left(\frac{1}{B}+\frac{1}{(\phi-1/M)N}\right)\log\frac{N}{M}\right)$$

- For a **constant $\boldsymbol{\alpha}$**, can support arbitrarily small thresholds φ with amortized cost

$$O\left(\frac{1}{B}\log\frac{N}{M}\right)$$

Takeaway: Online reporting comes at the cost of throughput but almost online reporting is essentially free!

Leveled External-Memory Reporting Table (LERT) [Pandey et al. SIGMOD ‘20]

- Given a stream of size N and φN > $\mathbf{\Omega(N/M)}$ the amortized cost of solving real-time event detection is

[illegible]

Can achieve timely reporting at effectively the optimal insert cost; no query cost

- [illegible] with amortized cost

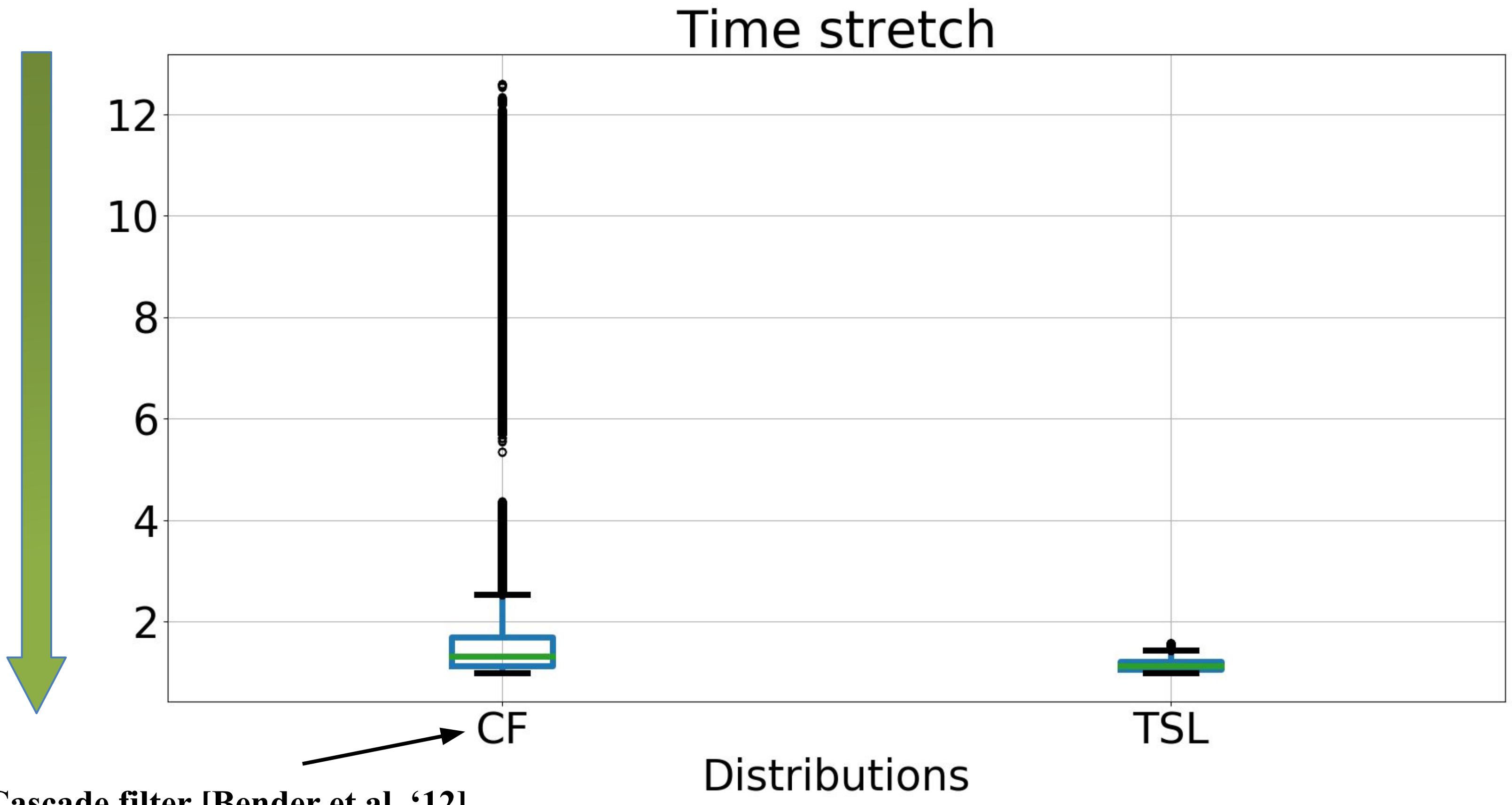
$$O\left(\frac{1}{B}\log\frac{N}{M}\right)$$

Takeaway: Online reporting comes at the cost of throughput but almost online reporting is essentially free!

Evaluation

- Empirical timeliness
- High-throughput ingestion

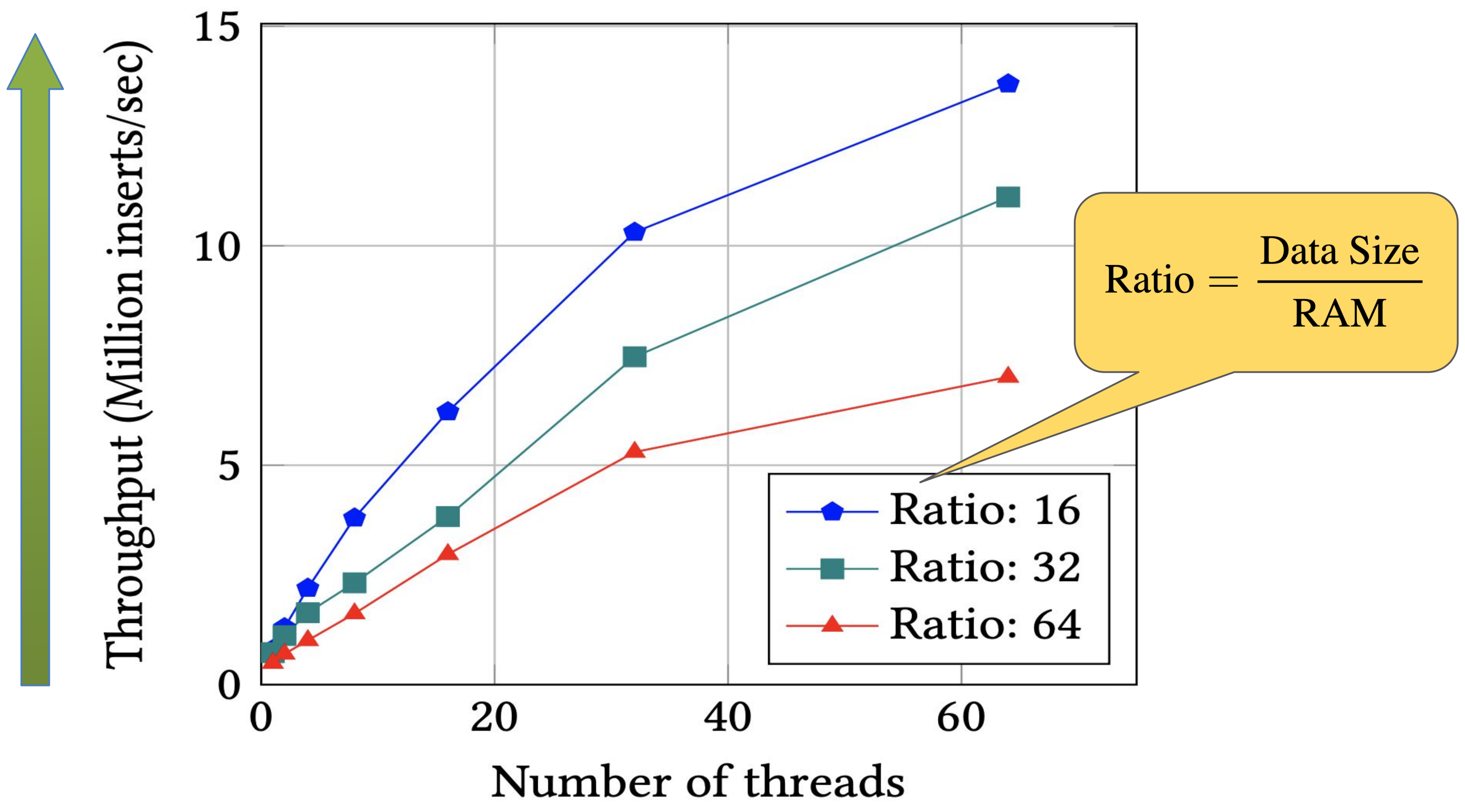
Evaluation: empirical time stretch



Cascade filter [Bender et al. '12]
State-of-the-art external memory counting table

Average time stretch is 43% smaller than theoretical upper bound.

Evaluation: scalability



The insertion throughput increases as we add more threads.

We can achieve > 13M insertions/sec.

LERT: supports scalable and real-time reporting

- Stream is large (e.g., terabytes) and high-speed (millions/sec)

High throughput ingestion ✓

- Events are high-consequence real-life events

No false-negatives; few false-positives ✓

Timely reporting (real-time) ✓

- Malicious traffic forms a small portion of the stream

Very small reporting thresholds ✓

Ongoing/future work

🟥	🟩	🟦
Shrink	Organize	Distribute

Data structures & Algorithms

- 🟥 **High performance filters/hash tables**
- 🟥 **Optimize large-scale indexes using ML**
- 🟥 **CQF on GPUs**

Systems

- 🟩 **Scalable streaming graph processing system**
- 🟩 **Event detection for infinite streams**
- 🟦 **Distributed hash table (based on quotient filters)**

Applications

Computational biology

- 🟩 **Population-scale index for multi-coordinate variation data**
- 🟦 **Graph neural networks for metagenomics**
- 🟦 **Distributed Mantis: publicly hosted for scientific use**

Ongoing/future work

Shrink | Organize | Distribute

Data structures & Algorithms

High performance filters/hash tables

Optimize large-scale indexes using ML

CQF on GPUs

Systems

Scalable streaming graph processing system

Distributed hash table (based on quotient filters)

Event detection for infinite streams

Applications

Computational biology

Population-scale index for multi-coordinate variation data

Graph neural networks for metagenomics

Distributed Mantis: publicly hosted for scientific use

Future work: next gen data structure library

Goal: Overcome ***decades-old*** data structure ***trade-offs*** using new algorithmic paradigms and modern hardware

Existing hash table techniques

Separate chaining

- Chaining with linked-list
- Chaining with binary tree

Open addressing

- Linear probing
- Coalesced chaining
- Cuckoo hashing
- Hopscotch hashing
- Robin Hood hashing
- 2-choice hashing
- d-left hashing

- Cuckoo hashing suffers from ***random hopping***
- Linear probing/Robin Hood hashing suffer from ***long chains***
- 2-choice/d-left hashing suffer from ***multiple probes***

Iceberg hash table

Collaborators: Joe Durie, Alex Conway, Rob Johnson, Michael Bender, Martin Farach-Colton

Single hashing

d-left hashing

Balanced for most items

Very low variance

- Step 1: set *primary bin* by single hashing
  - If the bin has < τ *type 1* items, insert the new item in the bin as *type 1*
- Step 2: If there are < □ *type 2* items, insert the new item using d-left as *type 2*
- Step 3: select the *primary bin* and inserted as *type 3*

$$\tau = h + (3h \log h)^{2/3} \quad \delta = cn \quad h = \frac{\#items}{\#bins}$$

Iceberg hash table

Collaborators: Joe Durie, Alex Conway, Rob Johnson, Michael Bender, Martin Farach-Colton

Single hashing

d-left hashing

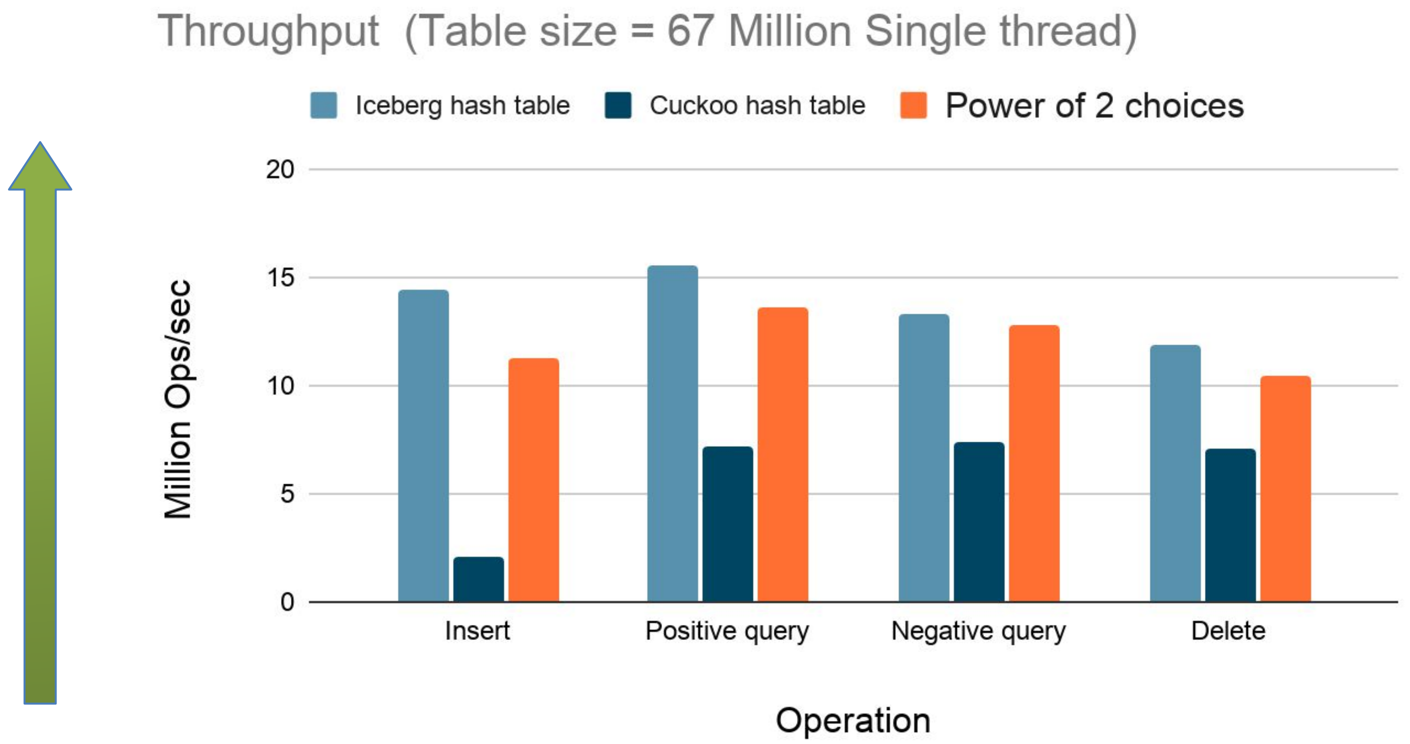
Limits variance across bins without random hopping and multiple probes

-
  - If the bin has < τ *type 1* items, insert the new item in the bin as *type 1*
- Step 2: If there are < □ *type 2* items, insert the new item using d-left as *type 2*
- Step 3: select the *primary bin* and inserted as *type 3*

$$\tau = h + (3h \log h)^{2/3} \quad \delta = cn \quad h = \frac{\#items}{\#bins}$$

Iceberg hash table performance

Collaborators: Joe Durie, Alex Conway, Rob Johnson, Michael Bender, Martin Farach-Colton



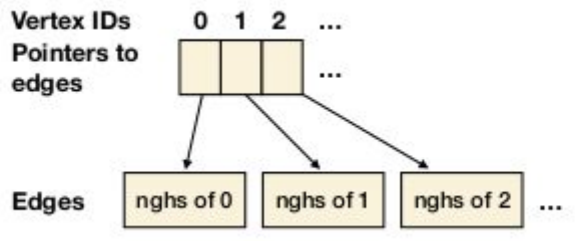
- **6.8X faster for insertions**
- **~2X faster for queries**
- **1.6X faster for deletes**

Future work: system for streaming graphs

Goal: build ***highly scalable*** streaming graph representation system

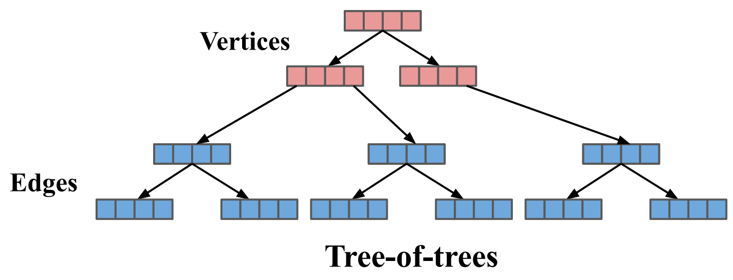
“One-size-fits-all” approach is suboptimal

Static



LIGRA [Shun & Blelloch ‘13]

Dynamic



ASPEN [Dhulipala et al. ‘19]

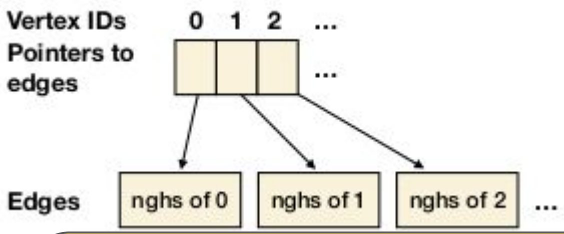
	LIGRA	ASPEN
add_edge	$O((\|E\| + \|V\|))/B)$	$O(\log \|V\| + c^2 \log(deg(u))/B)$
get_neighbors	$O(deg(u)/B)$	$O(\log \|V\| + deg(u)/B + deg(u)/c)$

Neighbor access requires at least ***two cache misses***

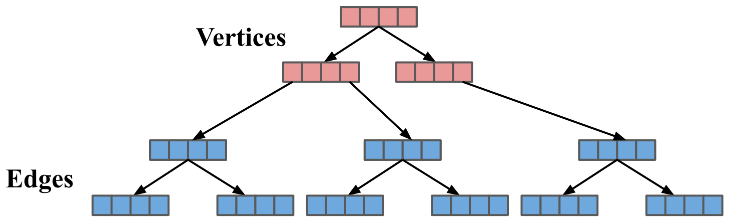
For dynamic, all operations have a ***log factor***

"One-size-fits-all" approach is suboptimal

Static



Dynamic



Static → Fast computations; no updates

Dynamic → Slower computations; updates

	LIGRA	ASPEN
add_edge	$O((\|E\| + \|V\|))/B)$	$O(\log \|V\| + c^2 \log(deg(u))/B)$
get_neighbors	$O(deg(u)/B)$	$O(\log \|V\| + deg(u)/B + deg(u)/c)$

Neighbor access requires at least ***two cache misses***

For dynamic, all operations have a ***log factor***

Real world graphs are often skewed

High variance in the degree distribution

Hierarchical structure + dynamic partitioning

Collaborators: Helen Xu, Brian Wheatman, Aydin Buluc, Kathy Yelick

High variance in the degree distribution

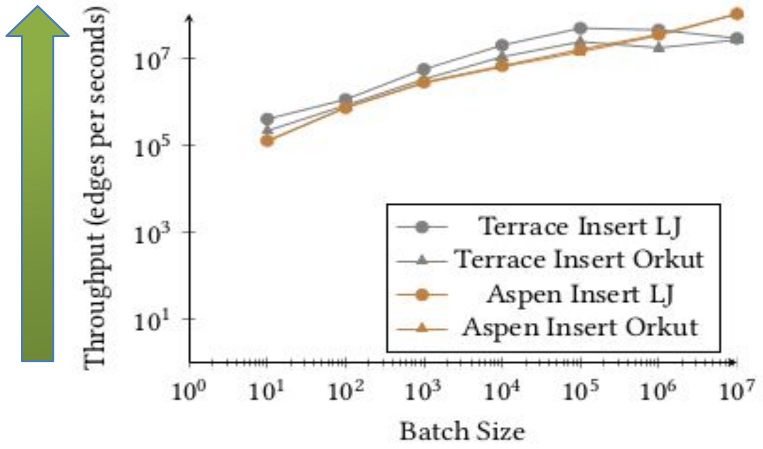
- In-place structure for vertices with low degree
- Shared sparse-array (PMA) for vertices with medium degree
- Independent B-tree for vertices with high degree

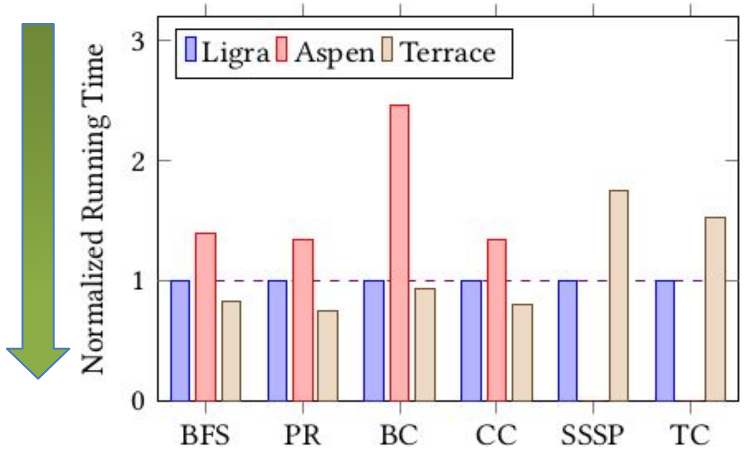
Hierarchical structure + dynamic partitioning

Collaborators: Helen Xu, Brian Wheatman, Aydin Buluc, Kathy Yelick

High variance in the degree distribution

- In-place structure for vertices with low degree
- Shared sparse-array (PMA) for vertices with medium degree
- Independent B-tree for vertices with high degree





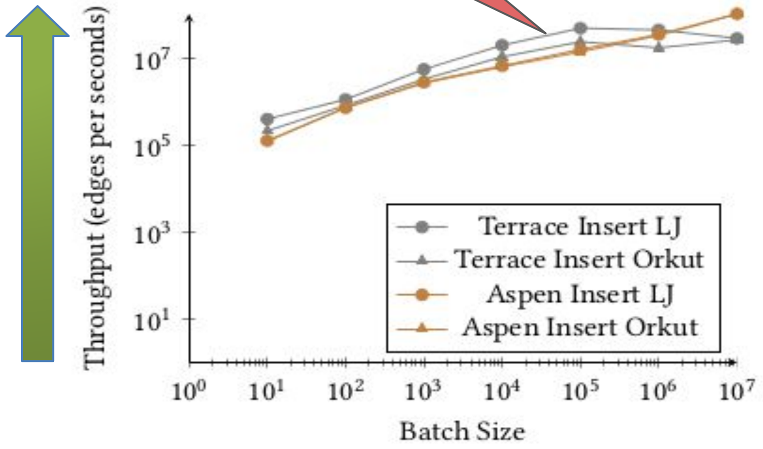
Hierarchical structure + dynamic partitioning

Collaborators: Helen Xu, Brian Wheatman, Aydin Buluc, Kathy Yelick

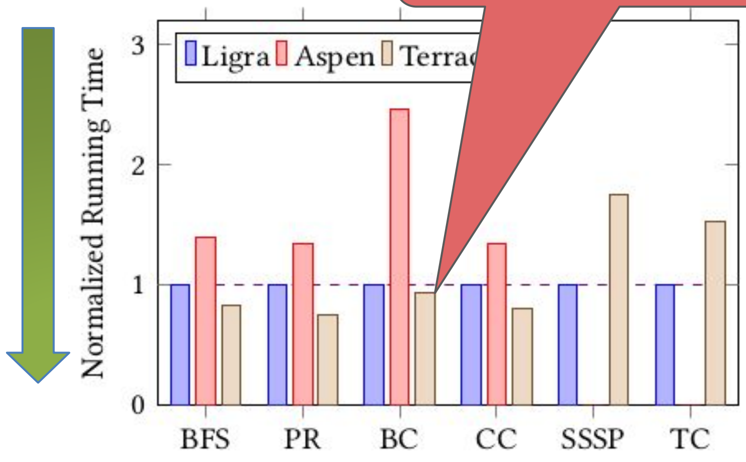
High variance in the degree distribution

- In-place structure for vertices with low degree
- Shared sparse-array (PMA) for vertices with medium degree
- Independent B-tree for vertices with high degree

Terrace: fast updates



Terrace: faster computations

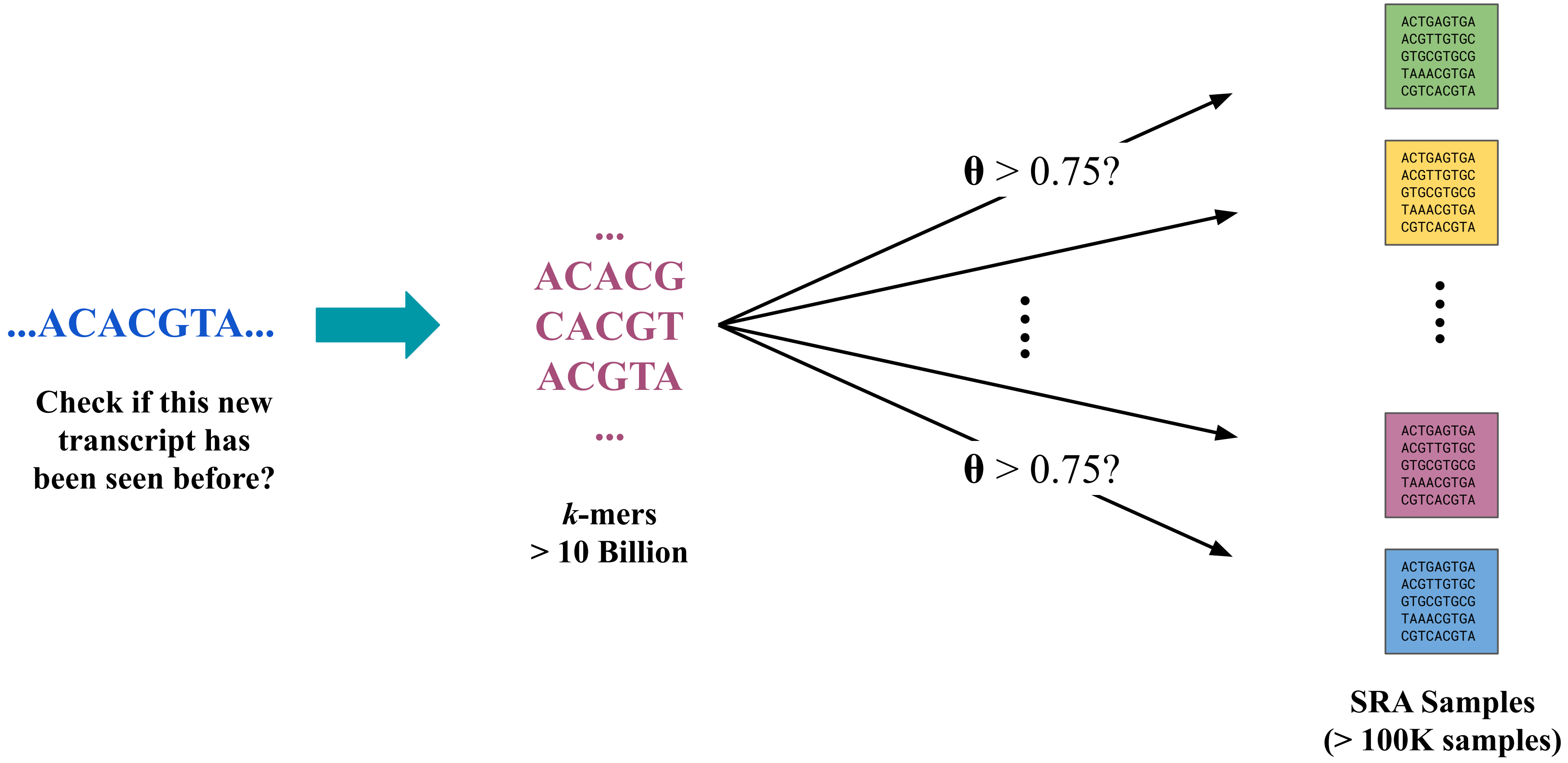


Future work: ML/DL for scalable indexing

Goal: optimize large-scale indexing solutions using machine learning

Sample discovery problem

Solomon & Kingsford Nat Biotech ‘16

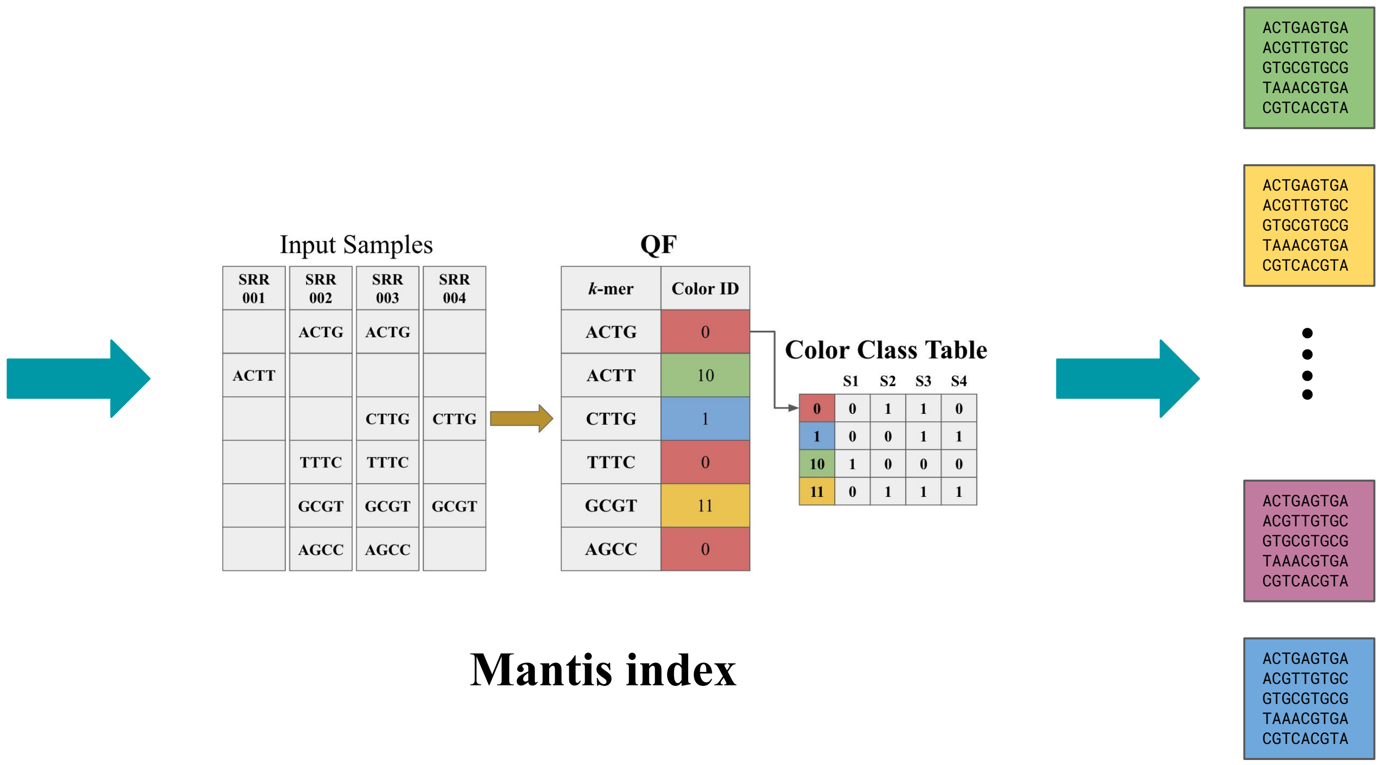


Return all samples that contain at least some user-defined fraction θ of the k-mers present in the query string

Mantis index for sample discovery problem

...ACACGTA...

Check if this new transcript has been seen before?



Input Samples

SRR 001	SRR 002	SRR 003	SRR 004
	ACTG	ACTG	
ACTT			
		CTTG	CTTG
	TTTC	TTTC	
	GCGT	GCGT	GCGT
	AGCC	AGCC	

QF

k-mer	Color ID
ACTG	0
ACTT	10
CTTG	1
TTTC	0
GCGT	11
AGCC	0

Color Class Table

	S1	S2	S3	S4
0	0	1	1	0
1	0	0	1	1
10	1	0	0	0
11	0	1	1	1

```
ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTGA
CGTCACGTA
```

```
ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTGA
CGTCACGTA
```

```
ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTGA
CGTCACGTA
```

```
ACTGAGTGA
ACGTTGTGC
GTGCGTGCG
TAAACGTGA
CGTCACGTA
```

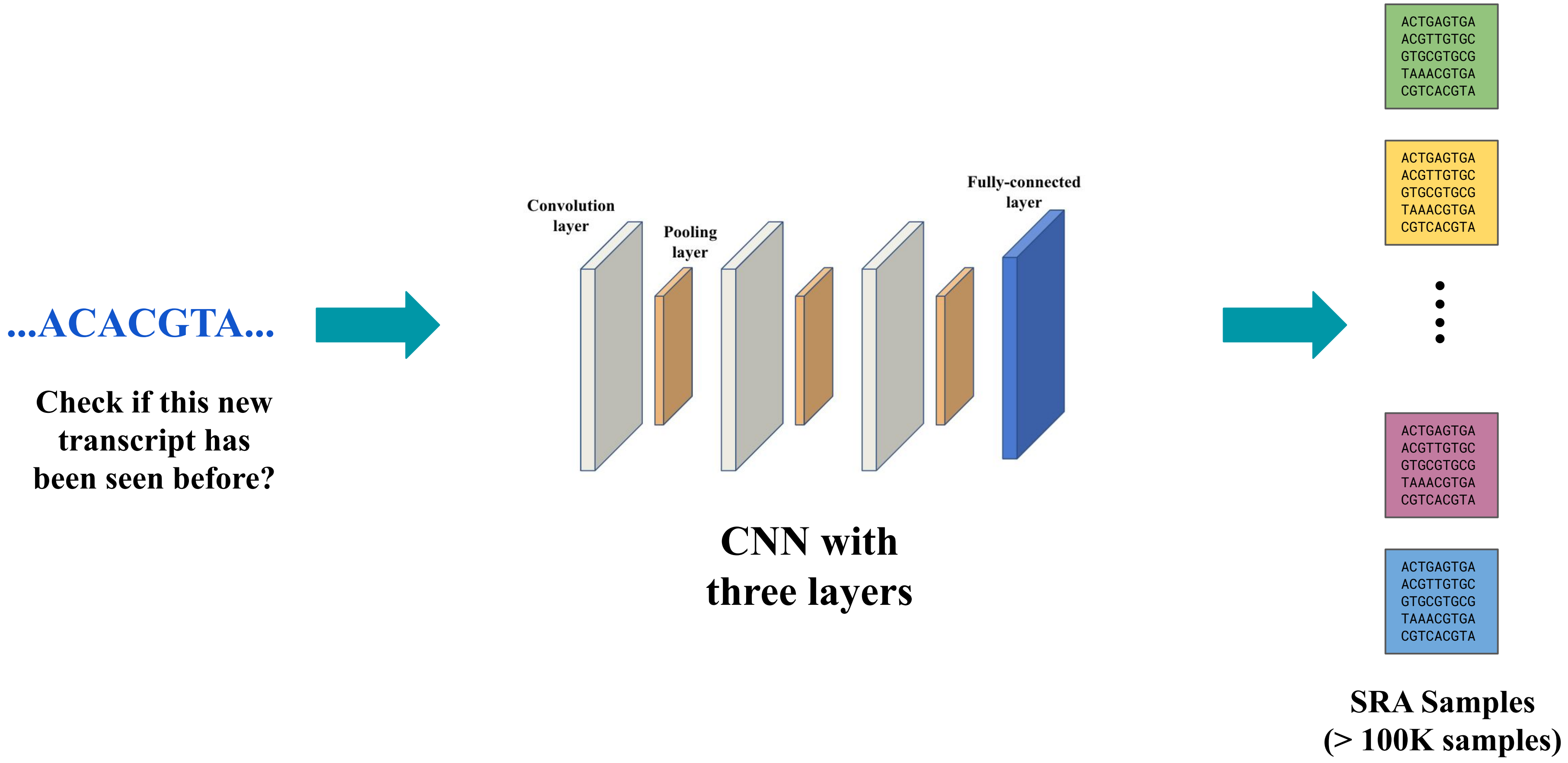
Mantis index

SRA Samples (> 100K samples)

Mantis index is based on a map from *k*-mers to a list of samples where the *k*-mer appears.

ML for sample discovery problem

Collaborators: Nick Bhattacharya, Aydin Buluc, Kathy Yelick

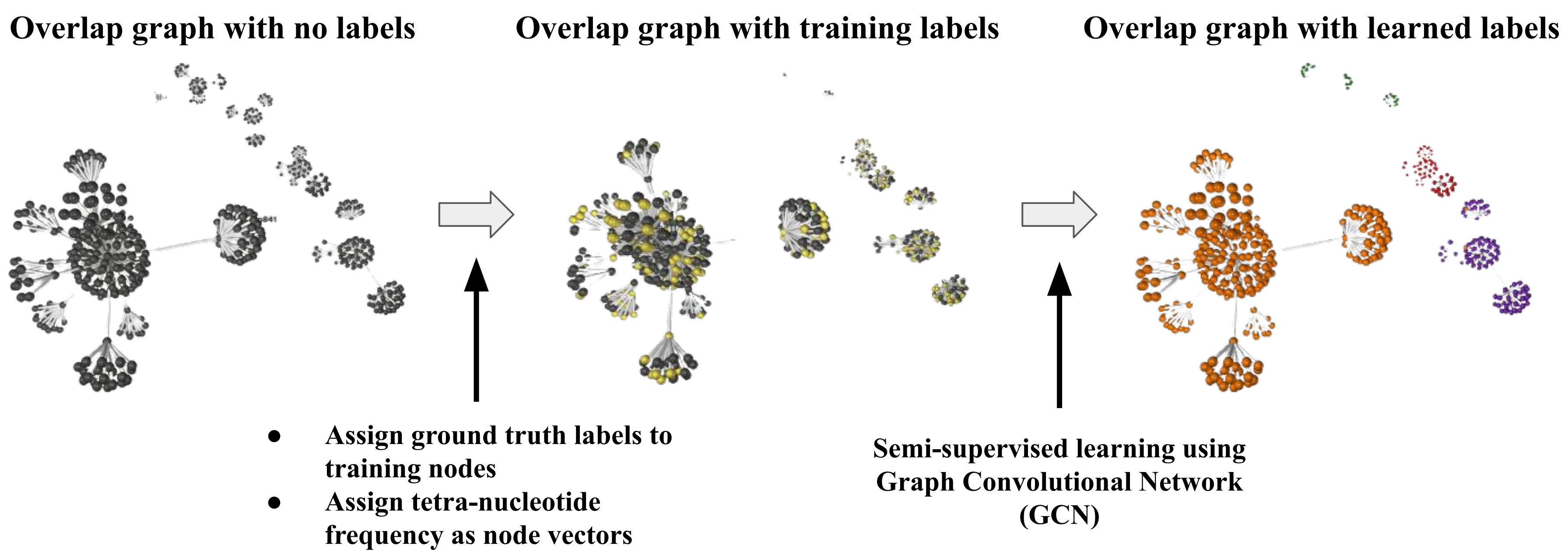

...ACACGTA...
Check if this new transcript has been seen before?
Convolution layer
Pooling layer
Fully-connected layer
CNN with three layers
ACTGAGTGA ACGTTGTGC GTGCGTGCG TAAACGTGA CGTCACGTA
ACTGAGTGA ACGTTGTGC GTGCGTGCG TAAACGTGA CGTCACGTA
ACTGAGTGA ACGTTGTGC GTGCGTGCG TAAACGTGA CGTCACGTA
ACTGAGTGA ACGTTGTGC GTGCGTGCG TAAACGTGA CGTCACGTA
SRA Samples (> 100K samples)

The loss function is optimized for the edit distance between sequences

We are planning to use (Order Min Hash ISMB ‘19) a LSH for edit distance

Metagenomic reads classification

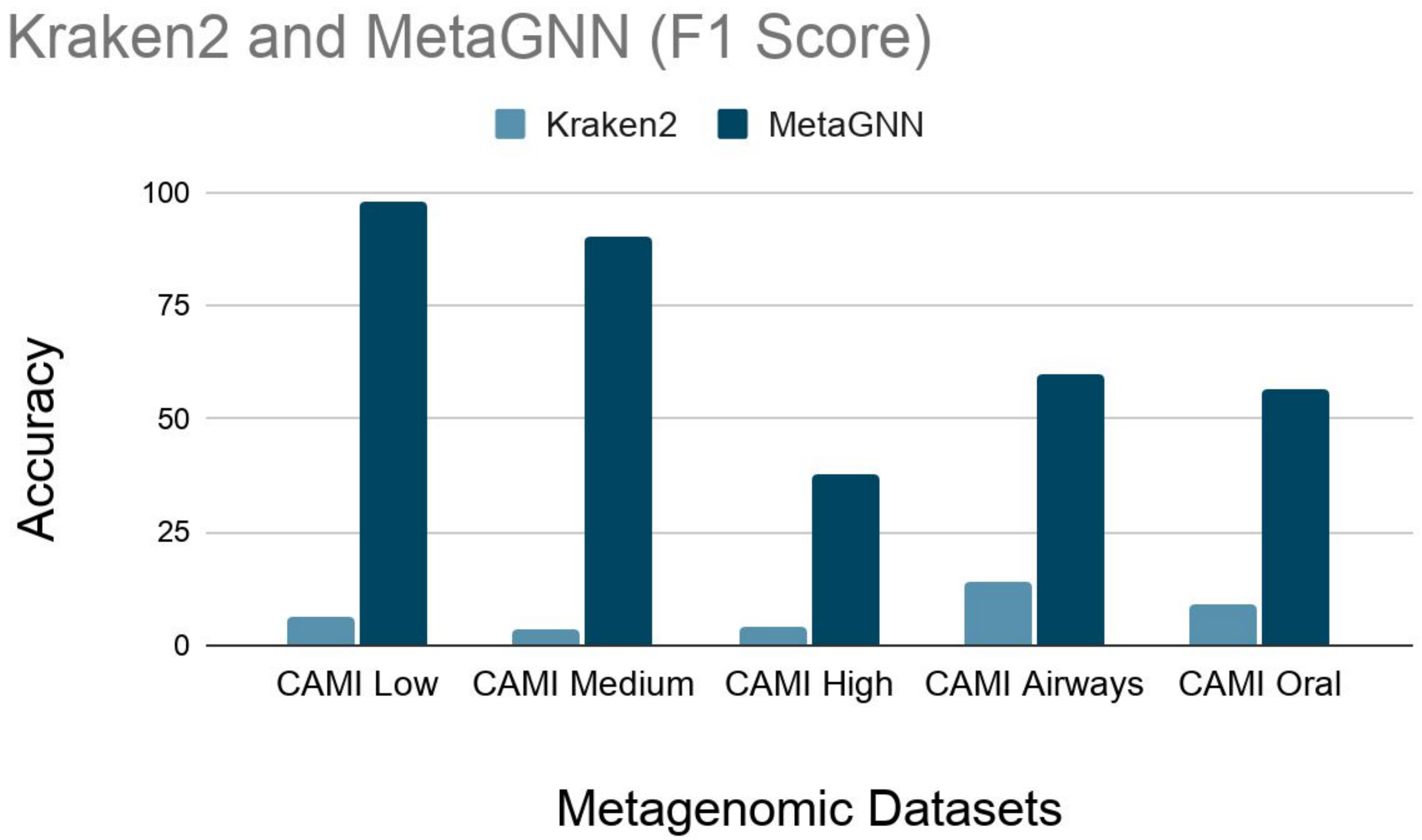
Collaborators: Giulia Guidi, Alok Tripathi, Aydin Buluc, Kathy Yelick


Overlap graph with no labels
Overlap graph with training labels
Overlap graph with learned labels
Assign ground truth labels to training nodes
Assign tetra-nucleotide frequency as node vectors
Semi-supervised learning using Graph Convolutional Network (GCN)

- Generate overlap graph: reads→nodes & overlap →edges
- Node features →Tetra nucleotide freq of reads
- Reference-based mapping as ground truth labels

Overlap graph + graph neural network (GNN)

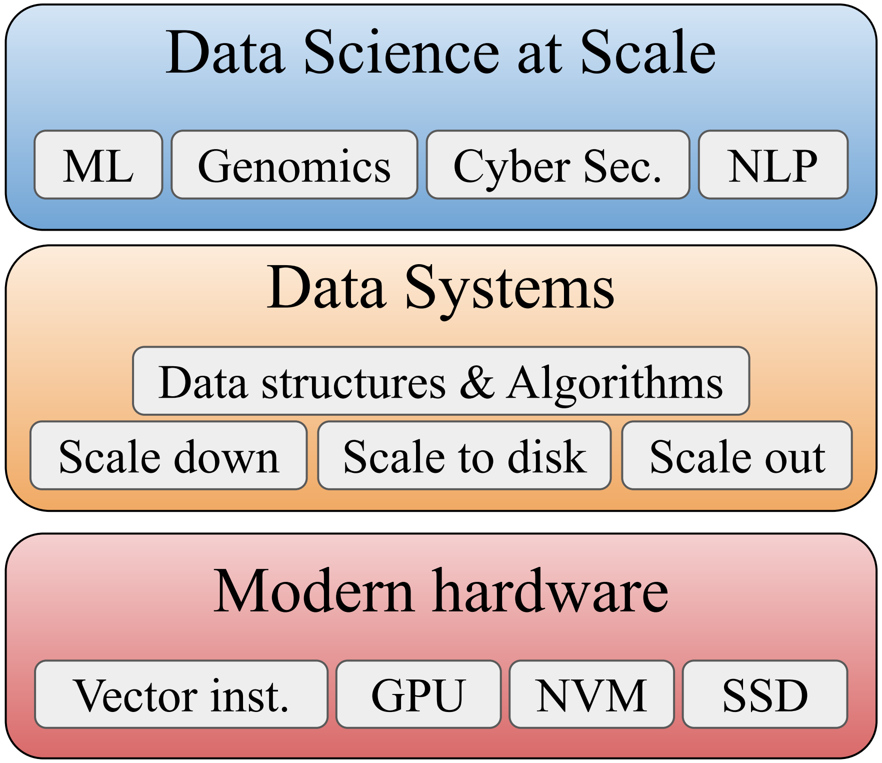
Collaborators: Giulia Guidi, Alok Tripathi, Aydin Buluc, Kathy Yelick



Can achieve higher accuracy using graph-based learning

Conclusion

- Scalability of data management systems will be the biggest challenge in future
- Changing hardware give rise to new algorithmic paradigms



We need to ***redesign*** existing data structures to take full advantage of modern hardware and ***rebuild*** data systems to efficiently support ***future*** data science.

https://prashantpandey.github.io

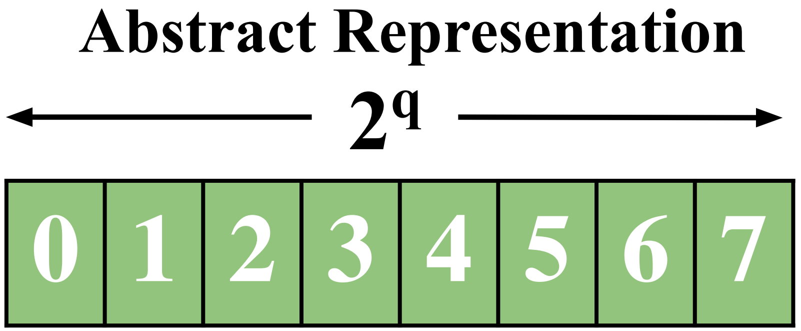
Backup slides

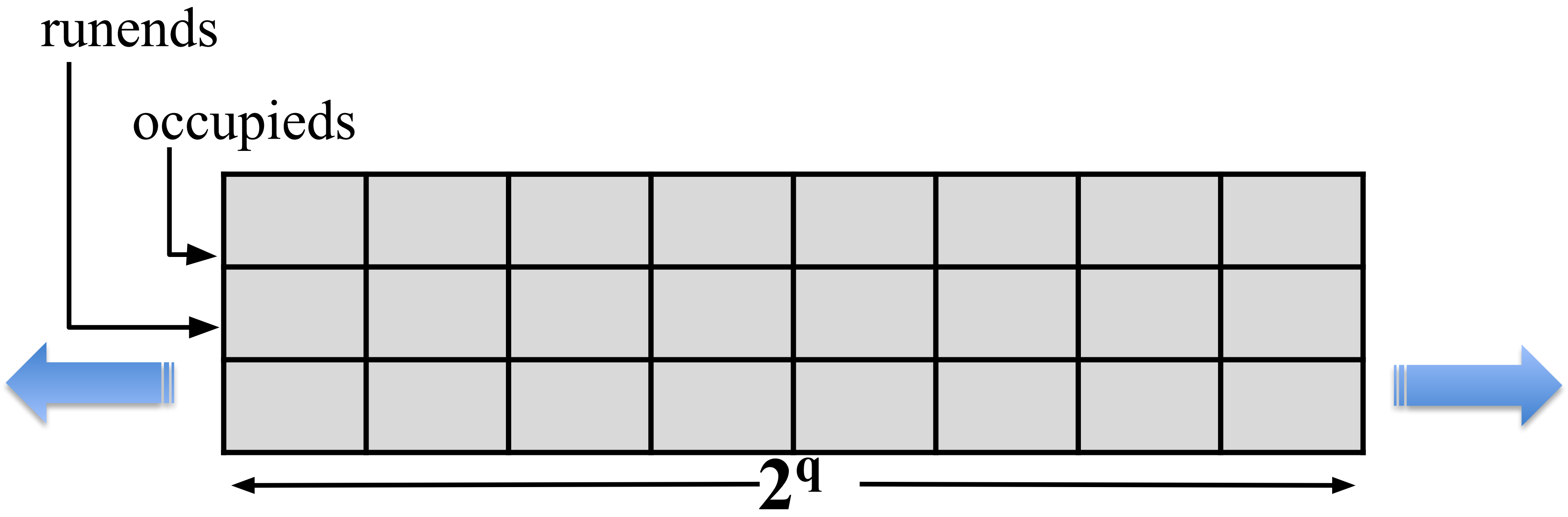
Quotient filter design

Implementation:

2 Meta-bits per slot.

h(x) --> $h_0(x) \,||\, h_1(x)$



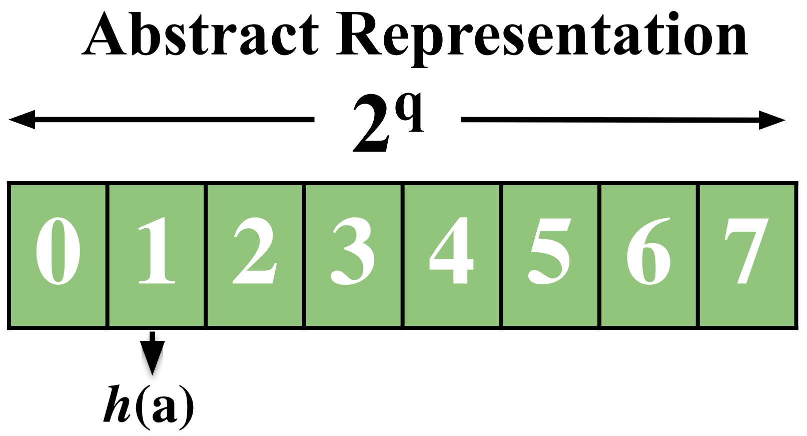


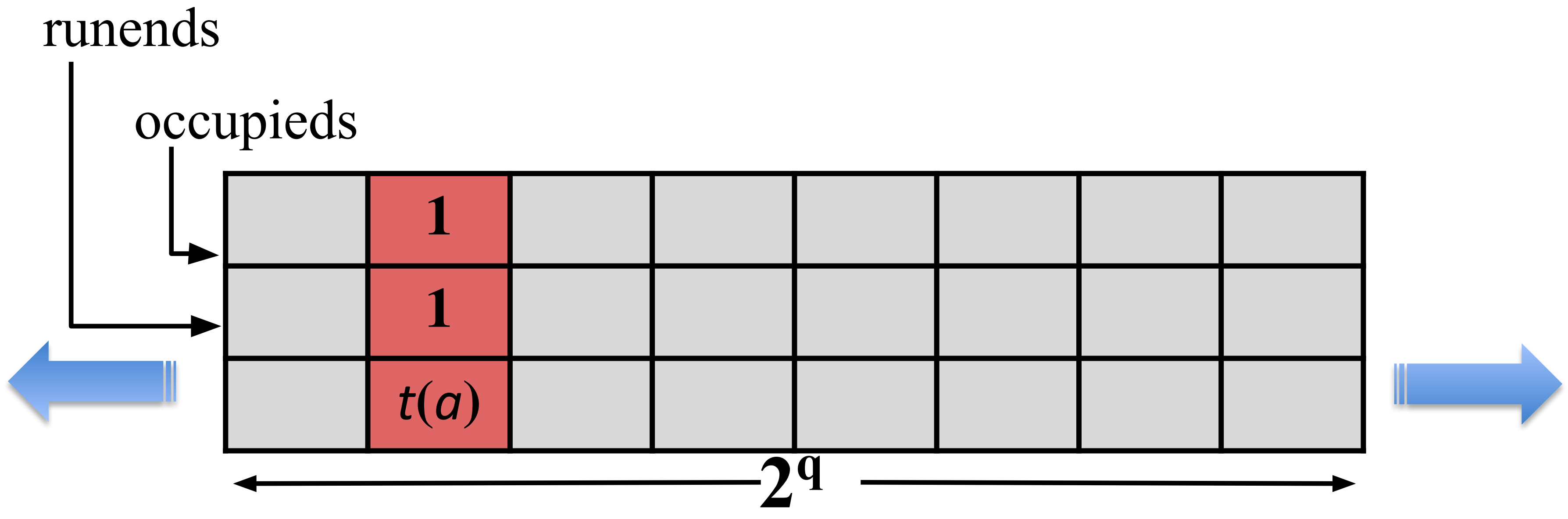
Quotient filter design

Implementation:

2 Meta-bits per slot.

h(x) --> h_0(x) || h_1(x)



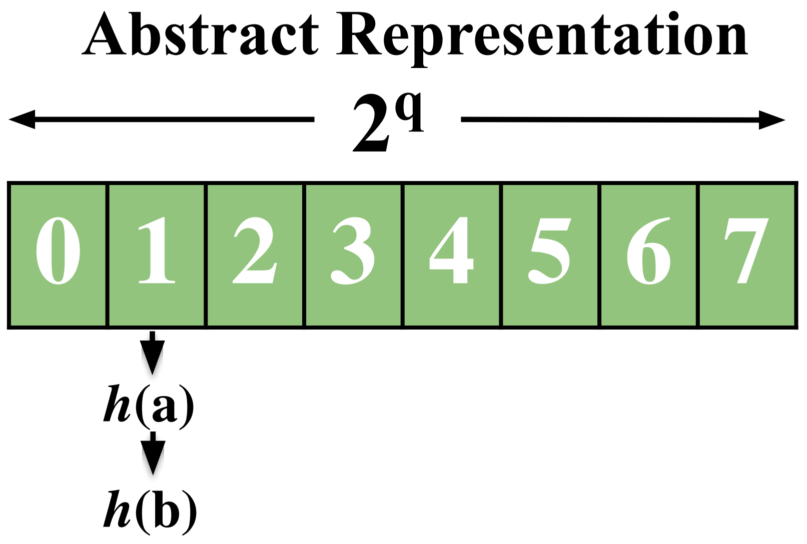


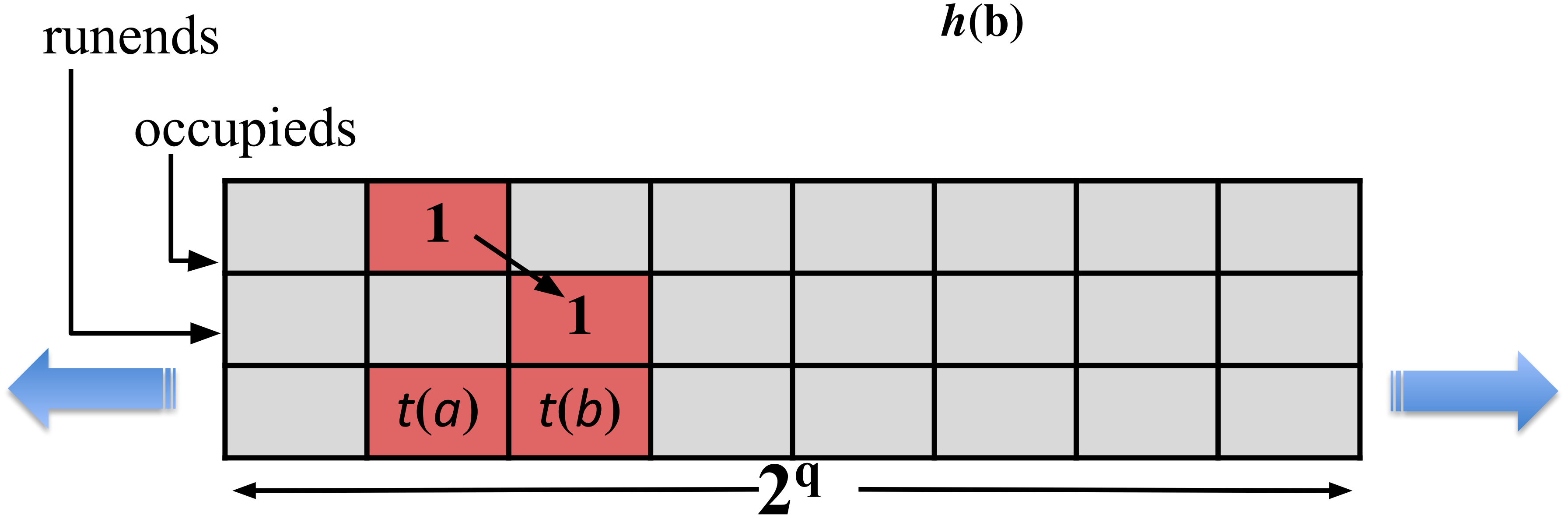
Quotient filter design

Implementation:

2 Meta-bits per slot.

h(x) --> h_0(x) || h_1(x)



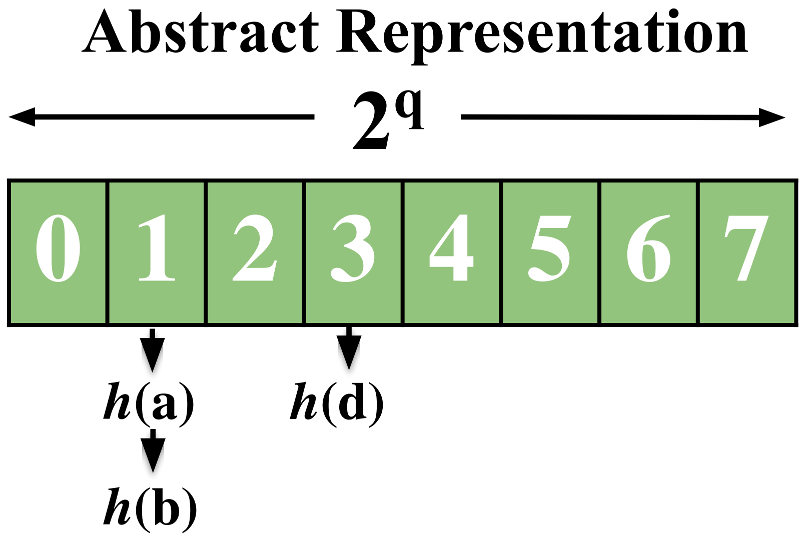


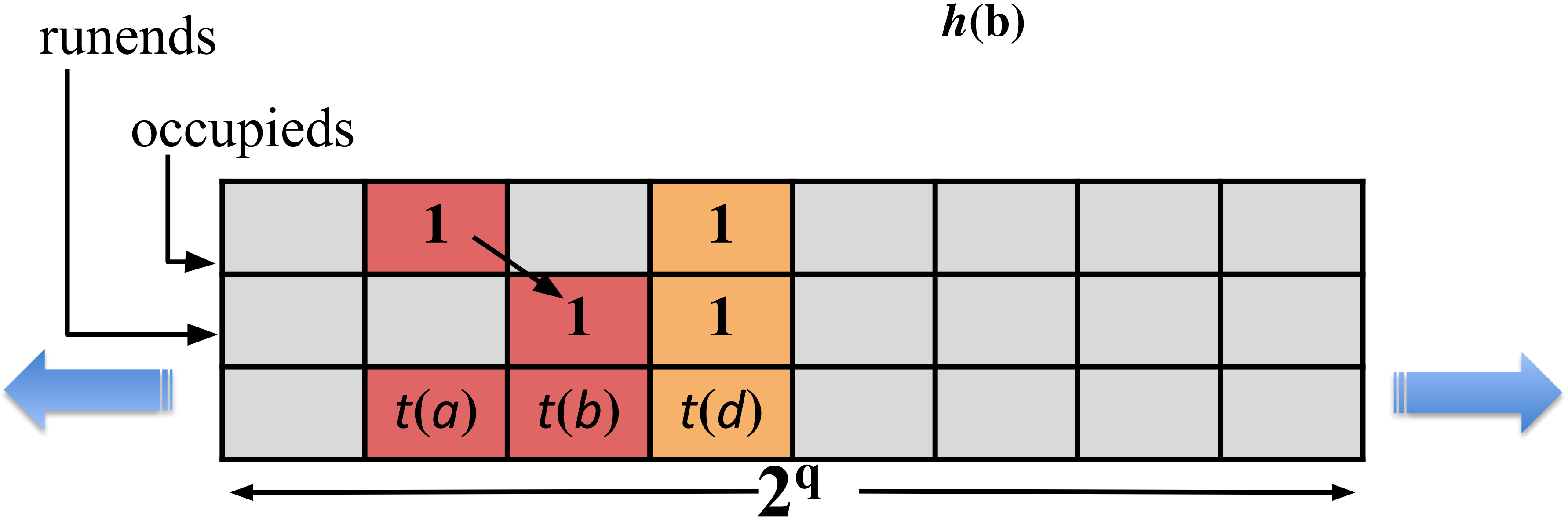
Quotient filter design

Implementation:

2 Meta-bits per slot.

$h(x) \dashrightarrow h_0(x) \,||\, h_1(x)$



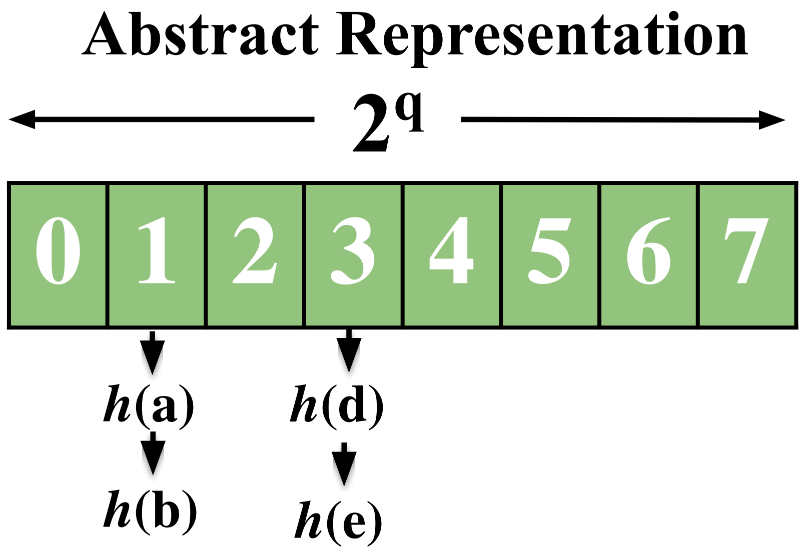


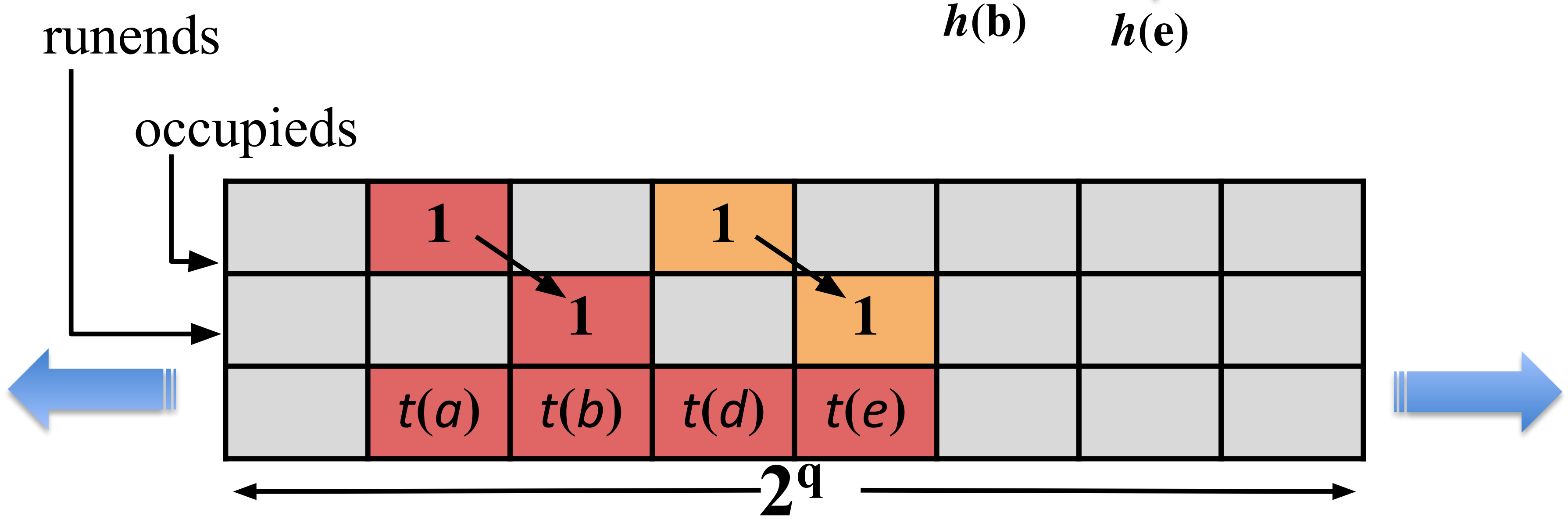
Quotient filter design

Implementation:

2 Meta-bits per slot.

h(x) --> h_0(x) || h_1(x)



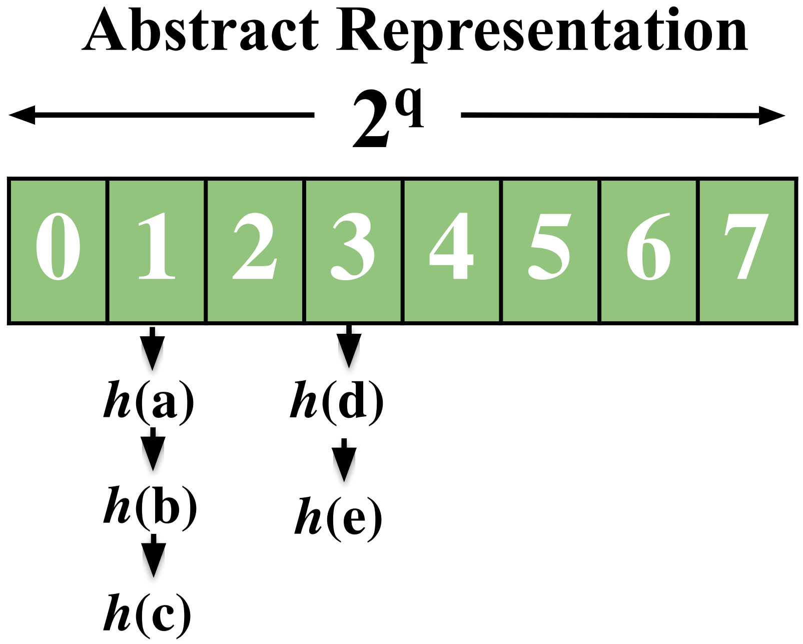


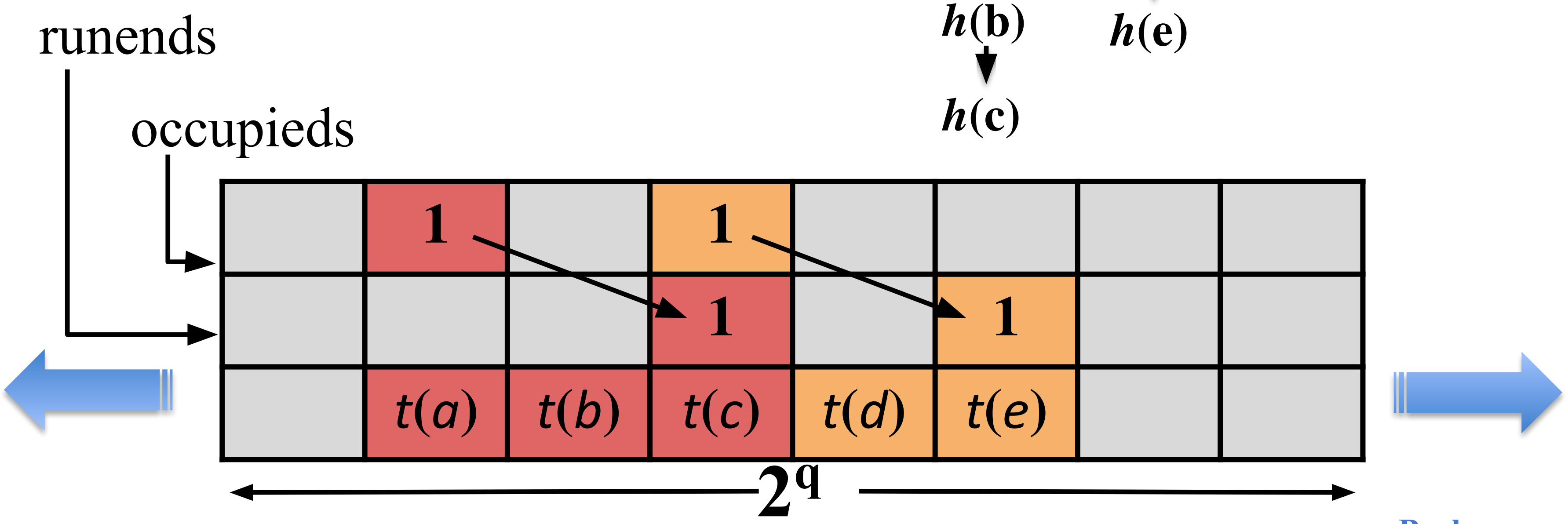
Quotient filter design

Implementation:

2 Meta-bits per slot.

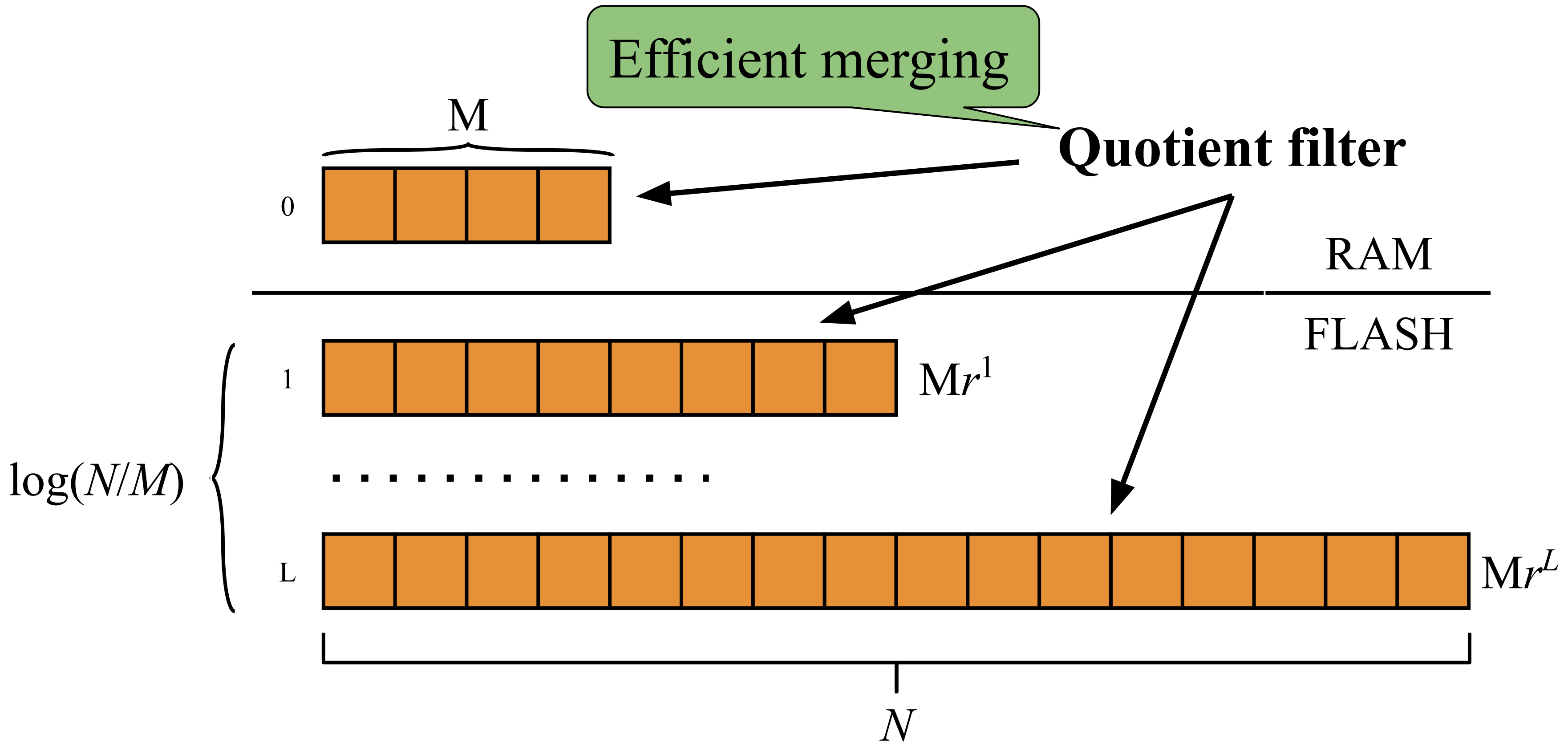
h(x) --> h_0(x) || h_1(x)





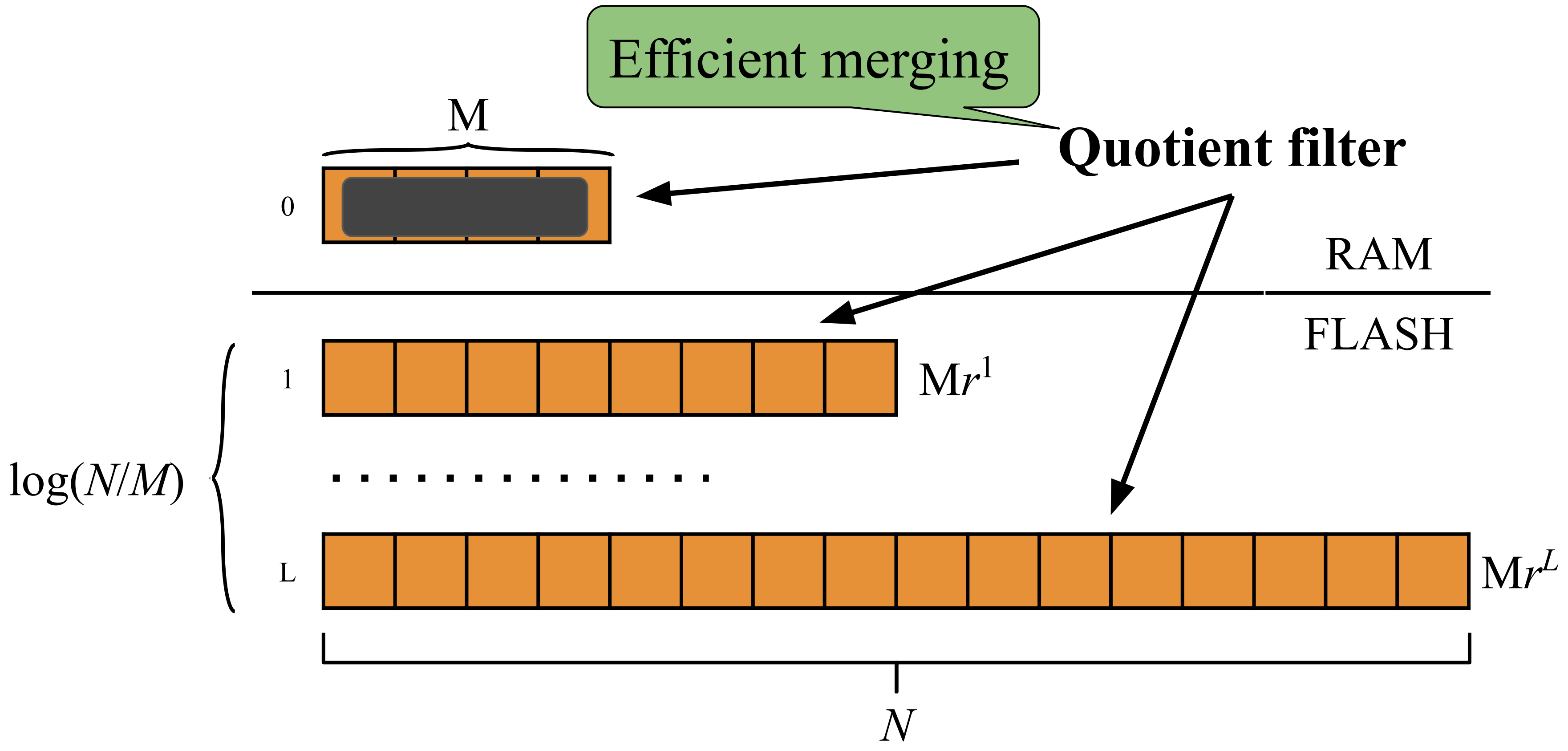
Back

Cascade filter: write-optimized quotient filter
[Bender et al. ‘12, Pandey et al. ‘17]



- The Cascade filter efficiently scales out-of-RAM
- It accelerates insertions at some cost to queries

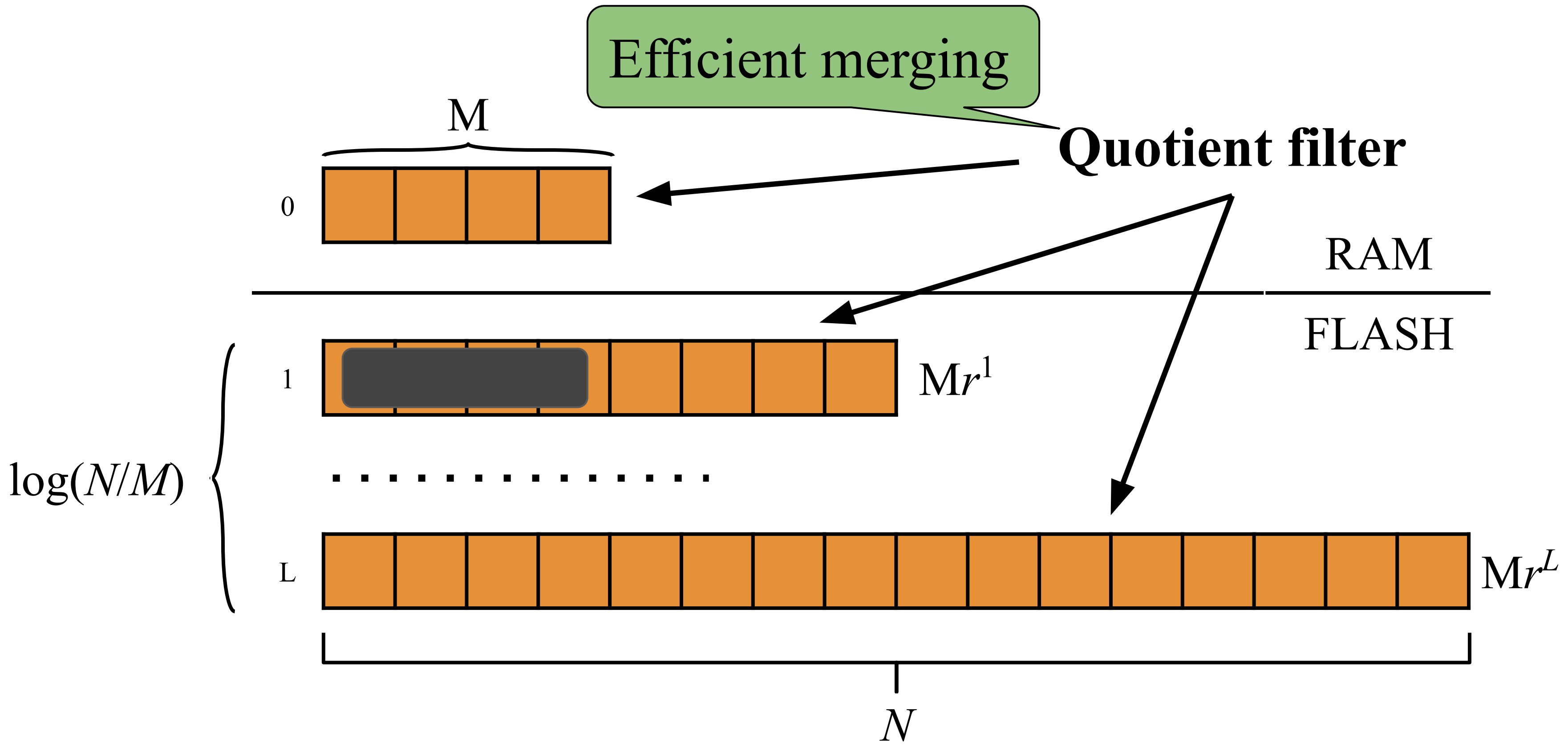
Cascade filter: flushing
[Bender et al. '12, Pandey et al. '17]



Items are initially inserted in the RAM level

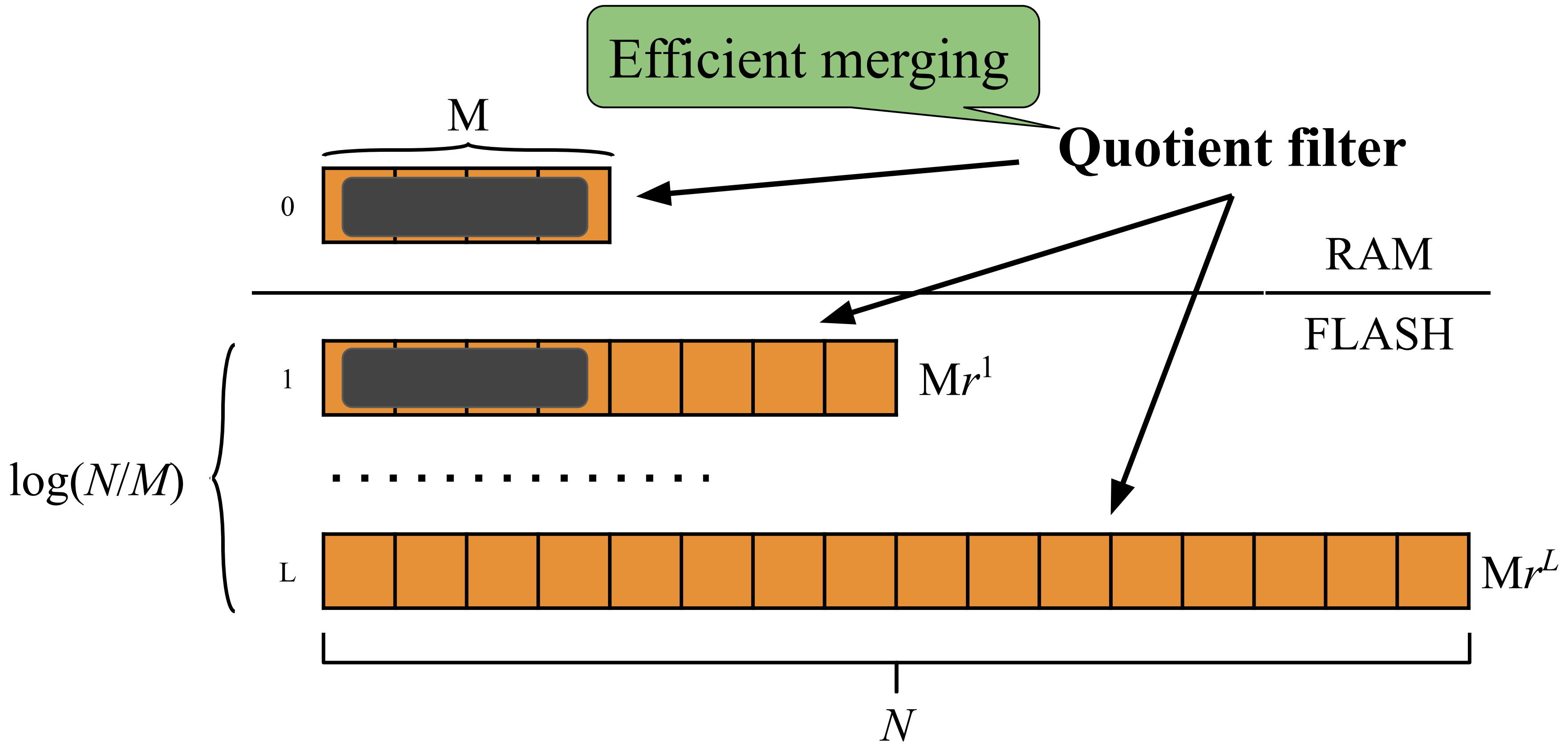
Cascade filter: flushing

[Bender et al. ‘12, Pandey et al. ‘17]



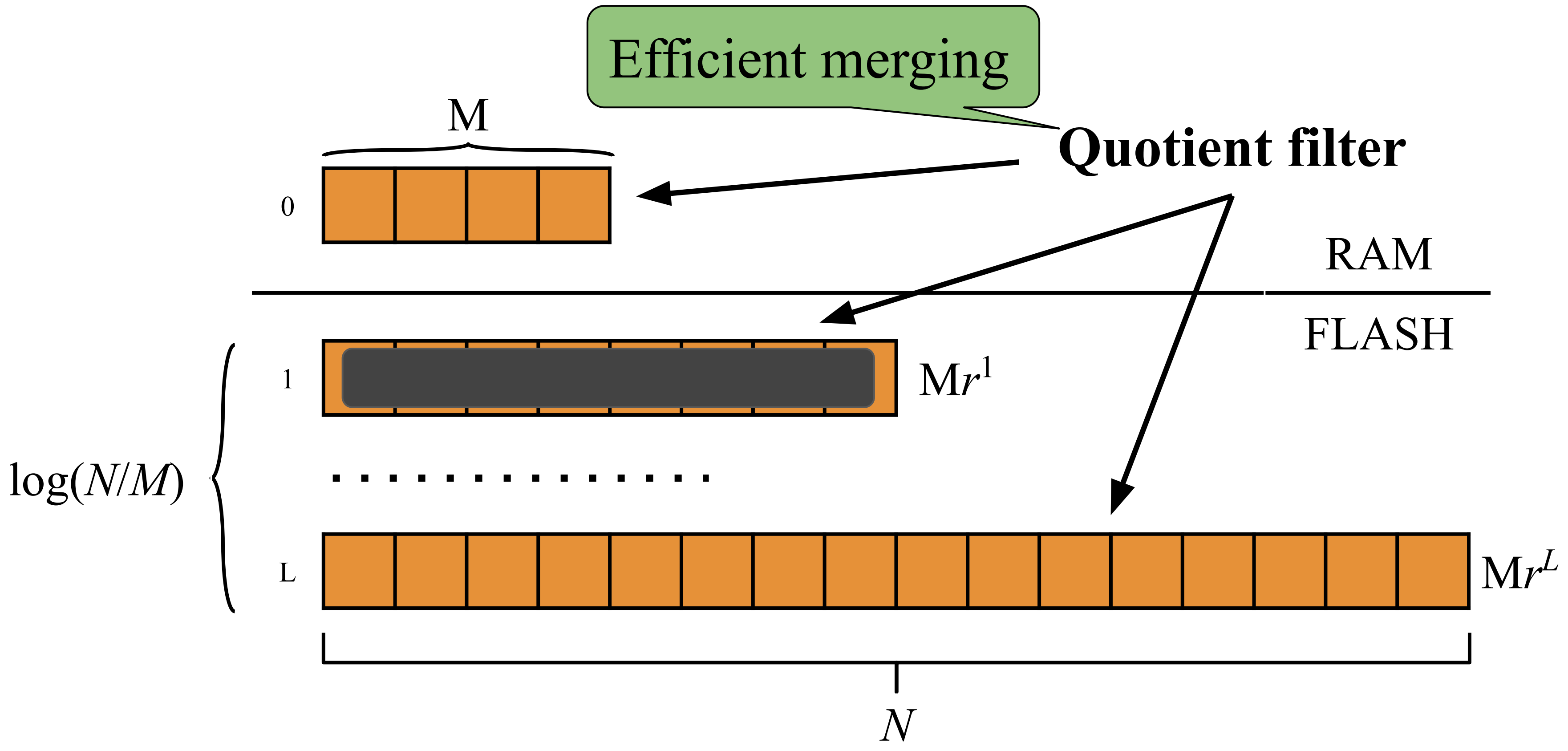
When RAM is full, items are flushed to the smallest level on disk ***i*** with space to insert items in level **0** to ***i*-1**

Cascade filter: flushing
[Bender et al. '12, Pandey et al. '17]



When RAM is full, items are flushed to the smallest level on disk ***i*** with space to insert items in level **0** to ***i*-1**

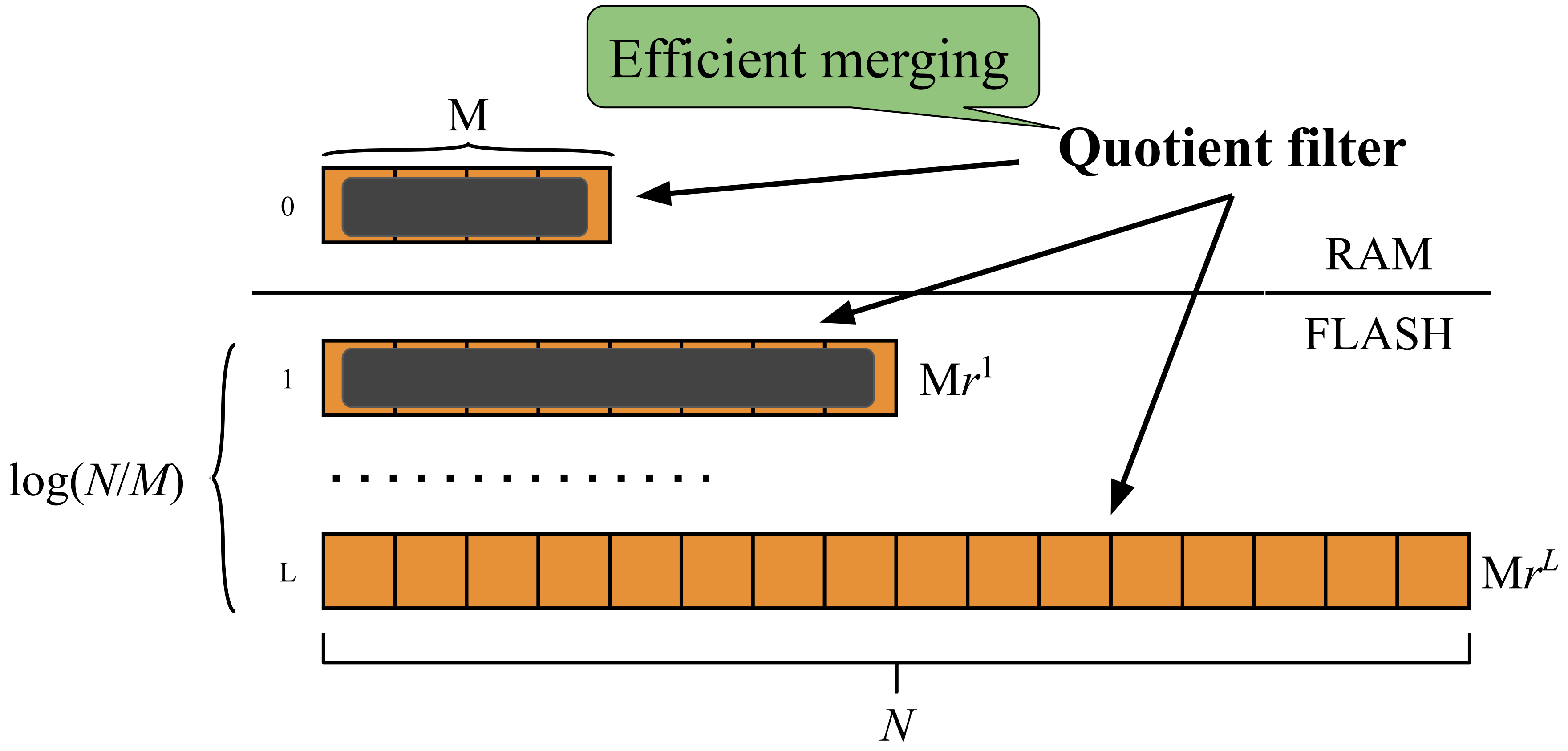
Cascade filter: flushing
[Bender et al. ‘12, Pandey et al. ‘17]



When RAM is full, items are flushed to the smallest level on disk ***i*** with space to insert items in level **0** to ***i*-1**

Cascade filter: flushing

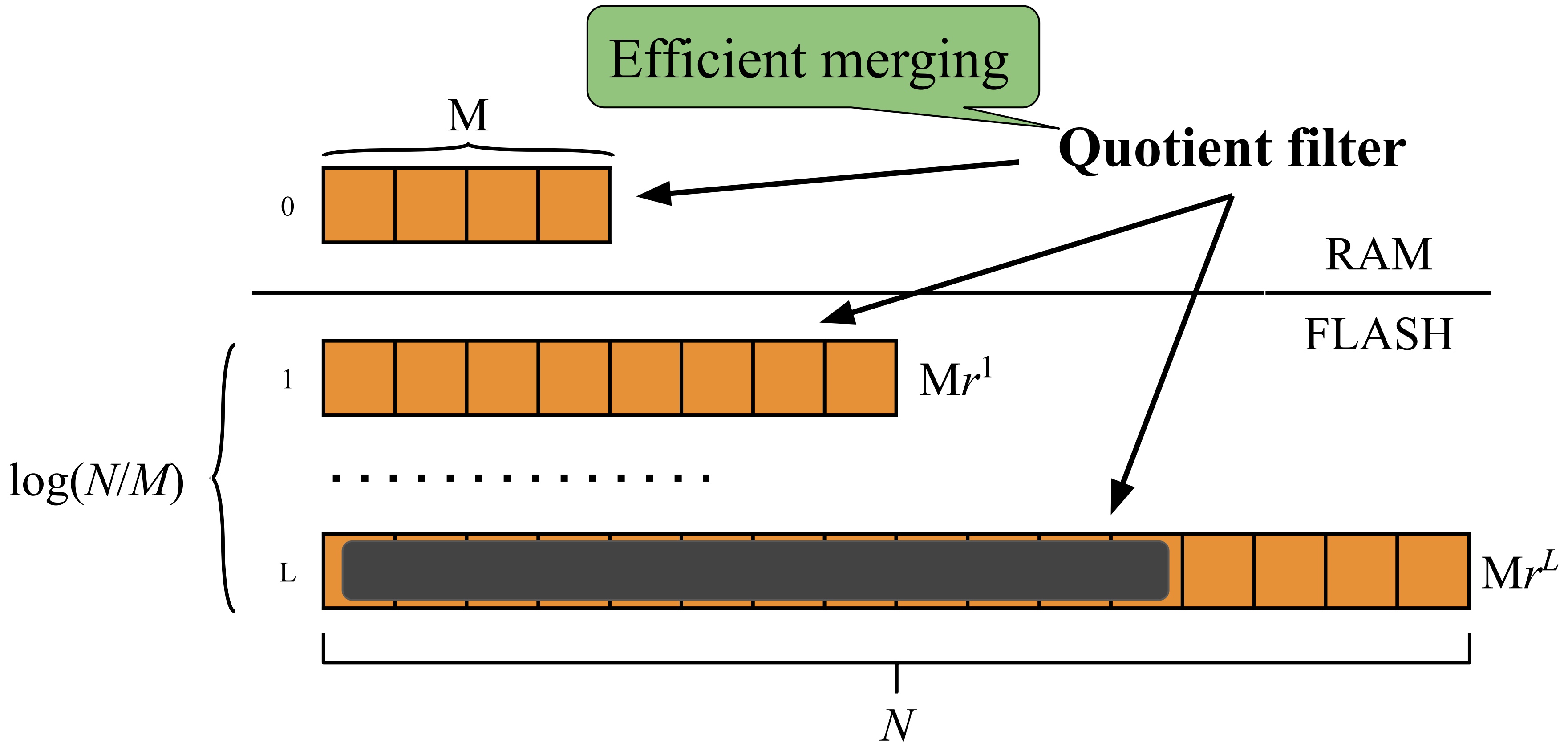
[Bender et al. ‘12, Pandey et al. ‘17]



When RAM is full, items are flushed to the smallest level on disk ***i*** with space to insert items in level **0** to ***i*-1**

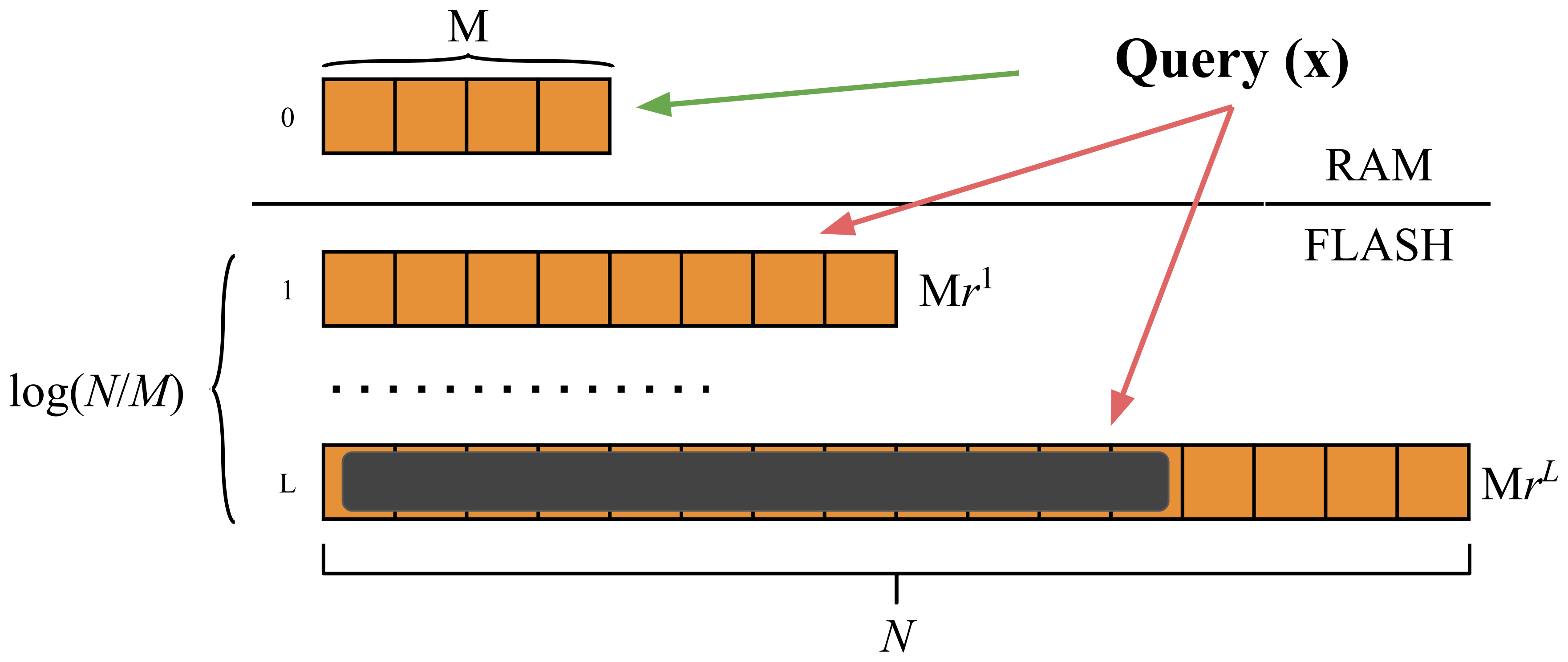
Cascade filter: flushing

[Bender et al. '12, Pandey et al. '17]



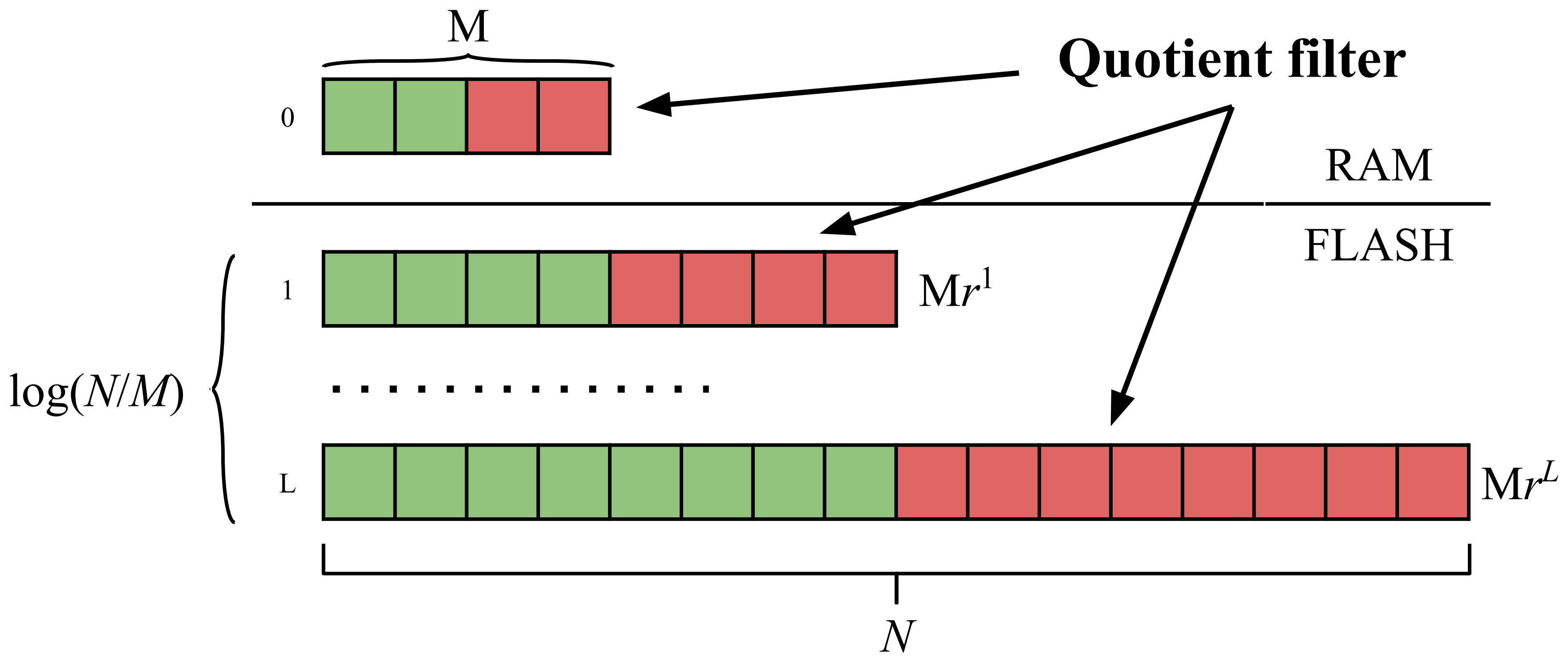
When RAM is full, items are flushed to the smallest level on disk ***i*** with space to insert items in level **0** to ***i*-1**

Cascade filter: query
[Bender et al. ‘12, Pandey et al. ‘17]



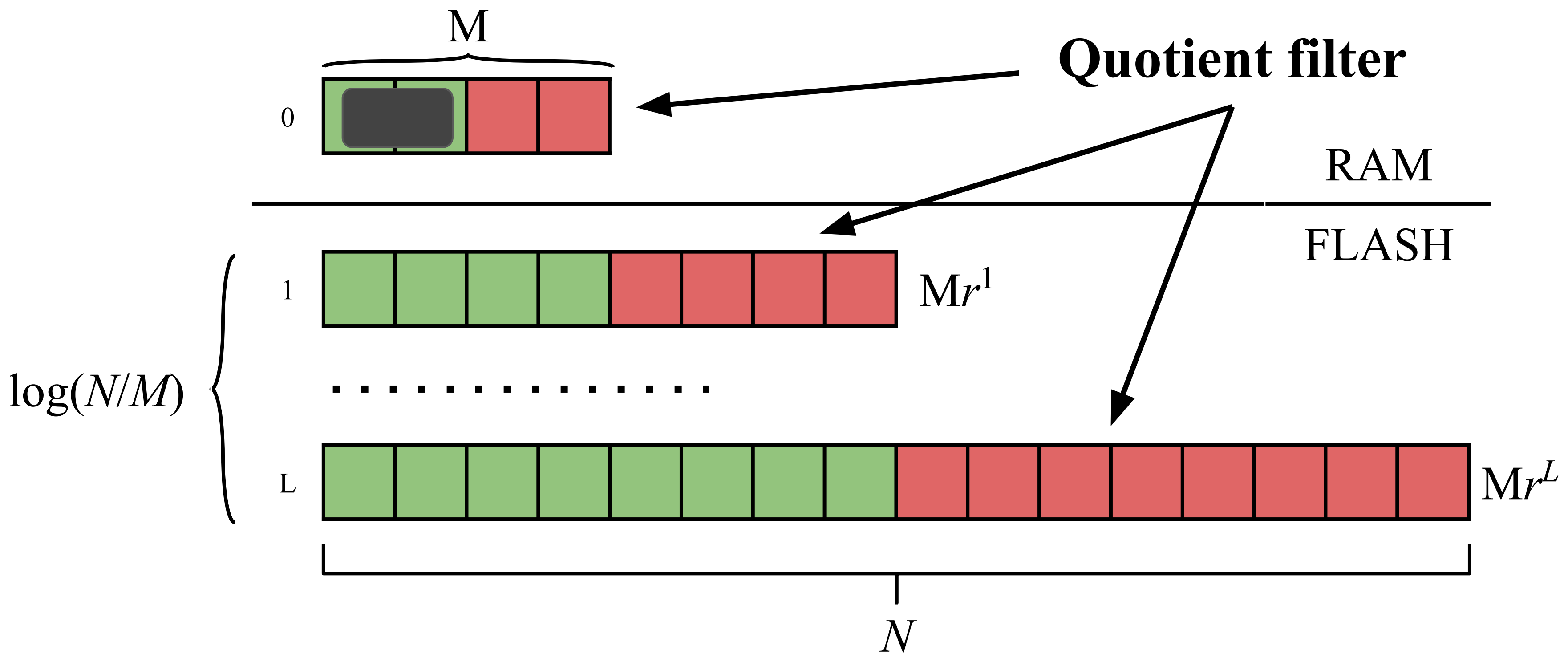
A query operation requires a lookup in each non-empty level

Time-stretch LERT



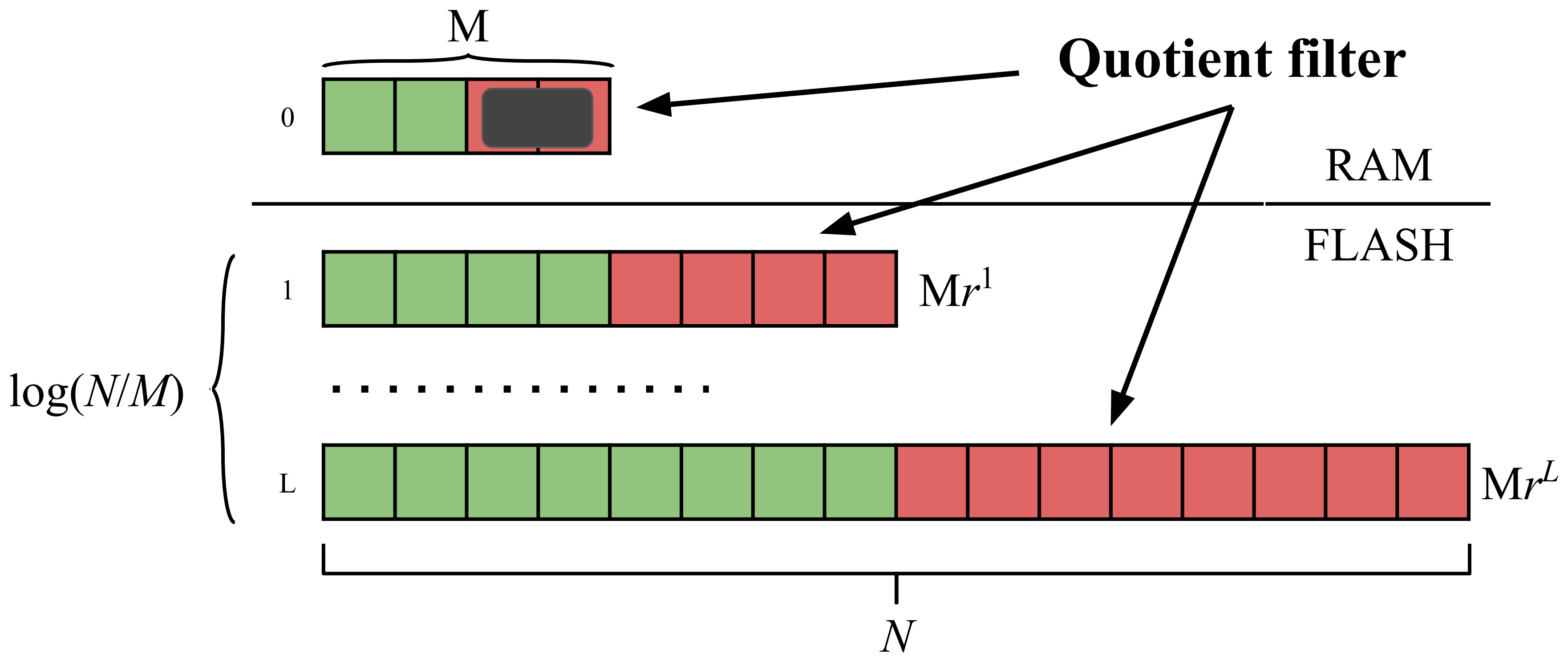
Divide each level into $1 + 1/\alpha$, equal-sized bins.

Time-stretch LERT



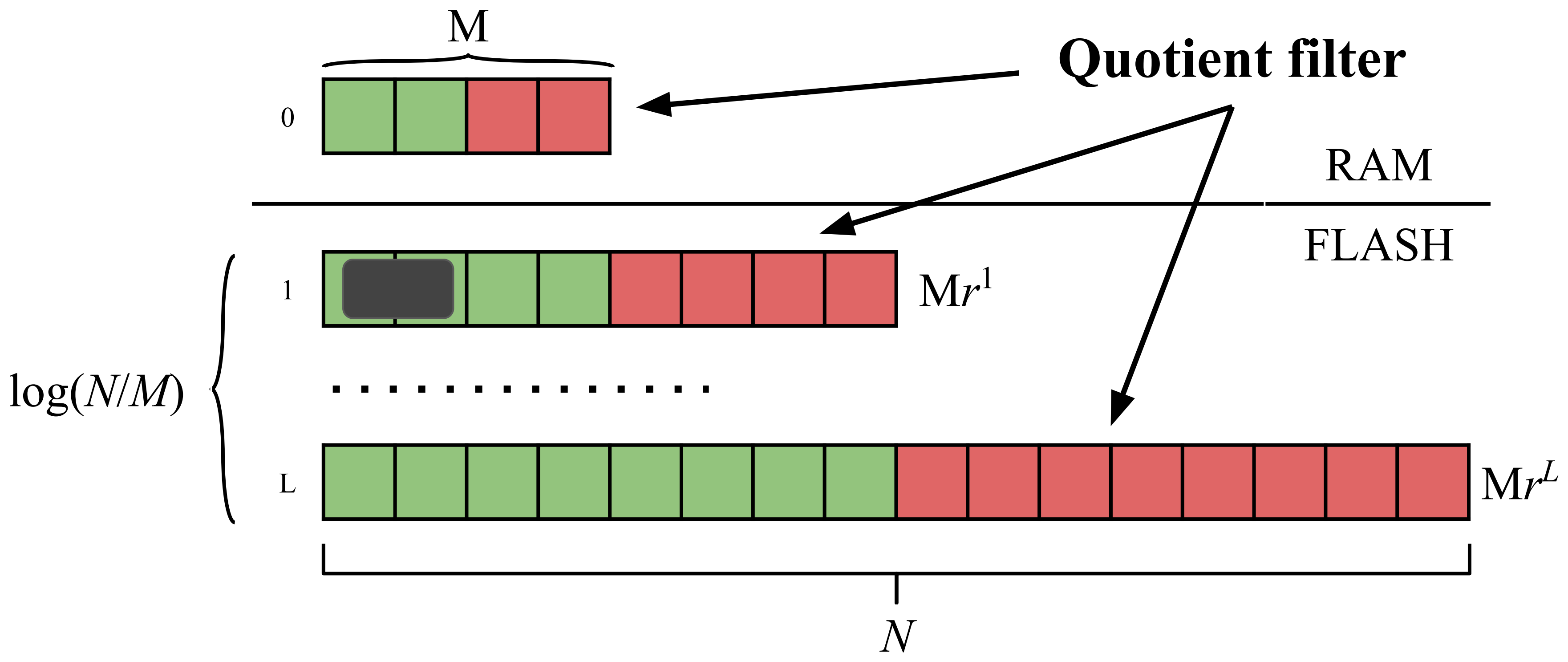
When a bin is full, items move to the adjacent bin

Time-stretch LERT



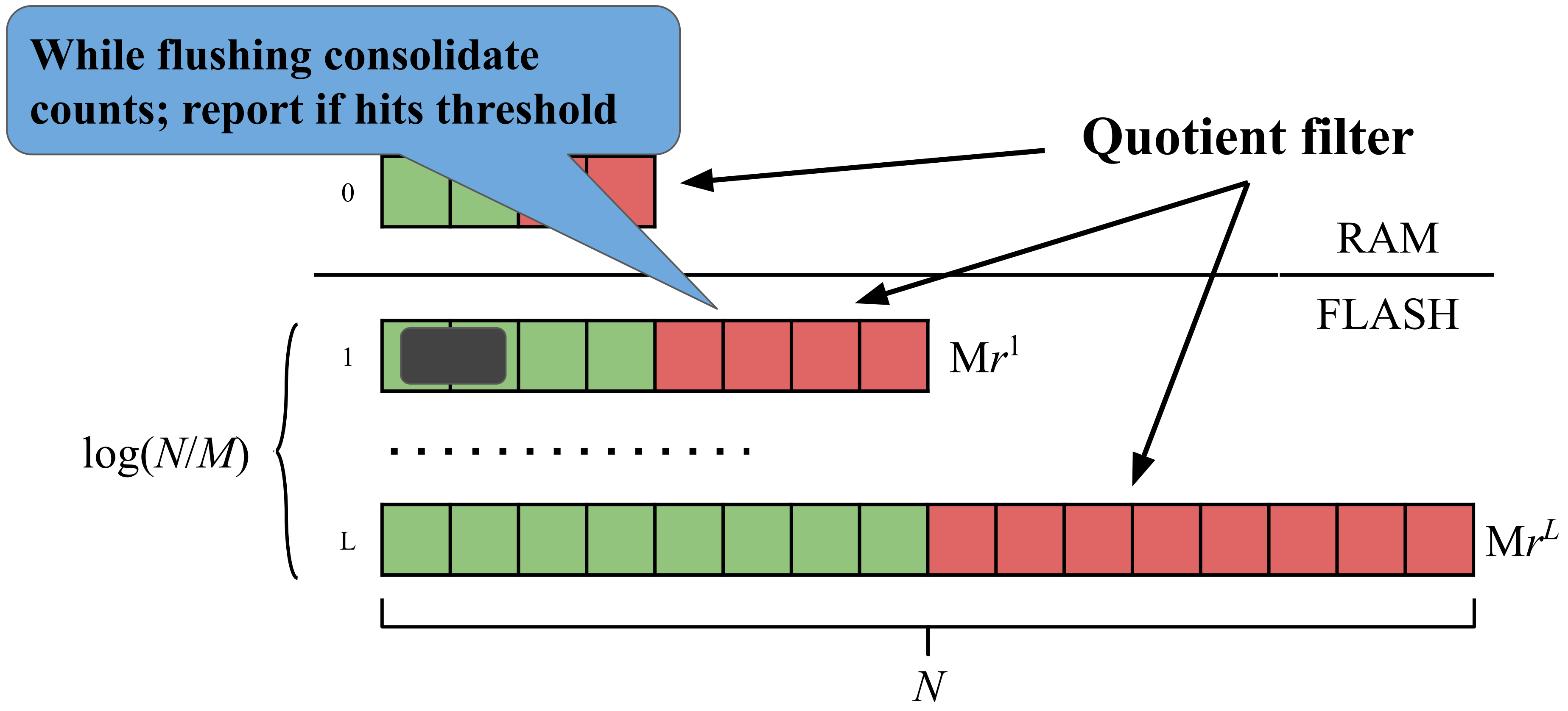
When a bin is full, items move to the adjacent bin

Time-stretch LERT



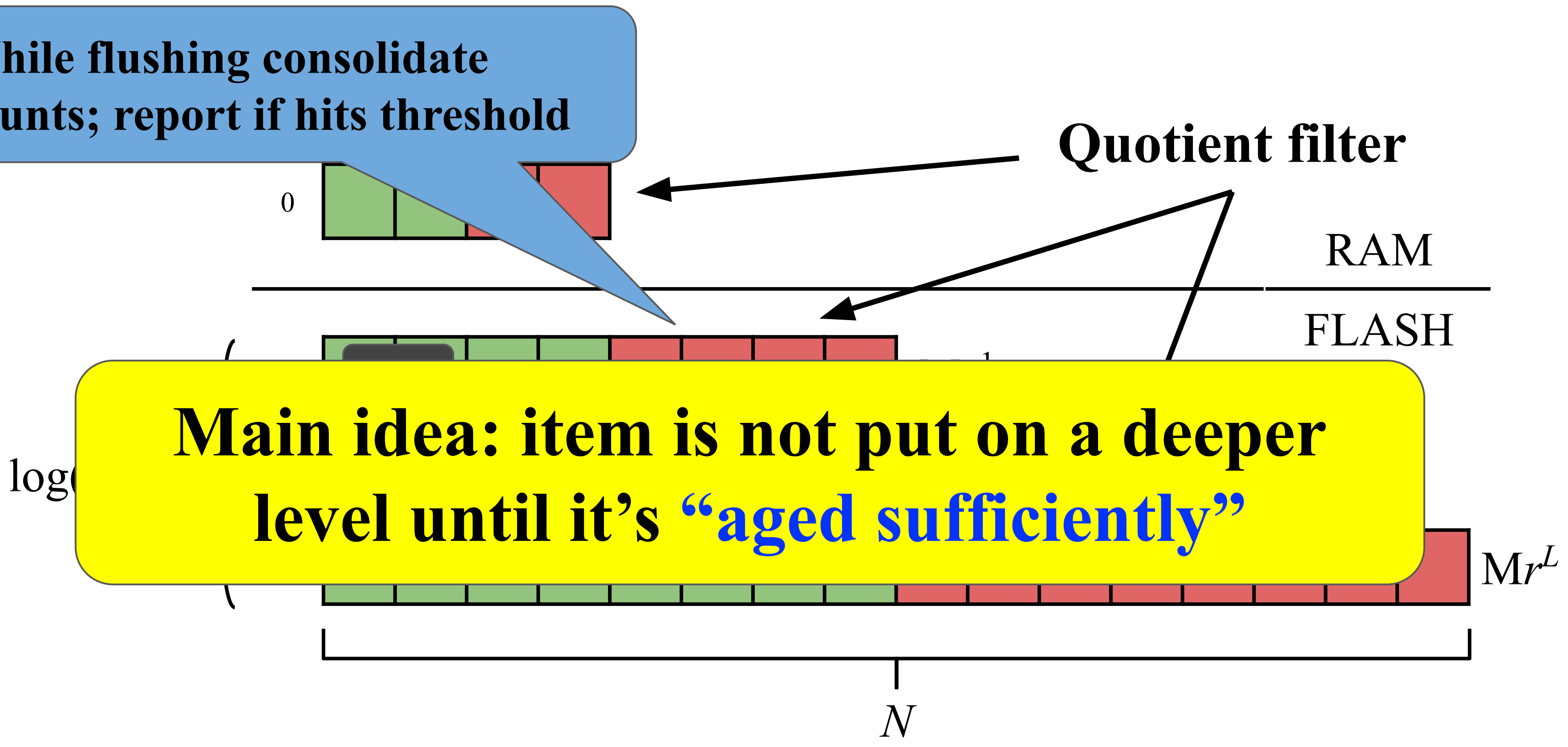
Last bin **flushed** to first bin of the next level

Time-stretch LERT



Last bin **flushed** to first bin of the next level

Time-stretch LERT



Main idea: item is not put on a deeper level until it's "aged sufficiently"

Last bin **flushed** to first bin of the next level

Time-stretch LERT I/O complexity

$$O\left(\left(\frac{\alpha+1}{\alpha}\right)\frac{1}{B}\log\frac{N}{M}\right)$$

Optimal insert cost for Write-optimized data structure

Time-stretch LERT I/O complexity

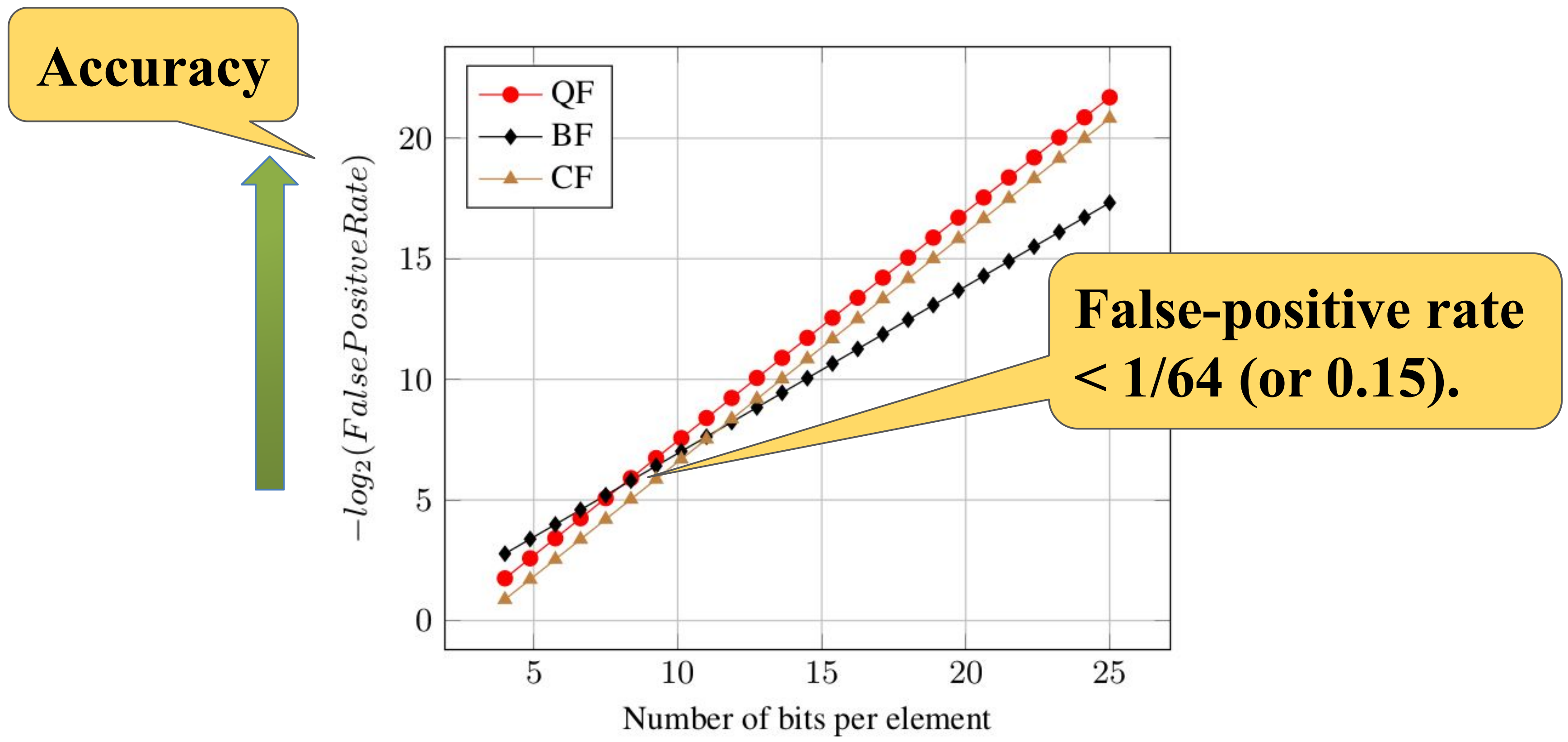
$$O\left(\left(\frac{\alpha+1}{\alpha}\right)\frac{1}{B}\log\frac{N}{M}\right)$$

Extra cost because we only move one bin during a flush. Constant loss for constant α

Optimal insert cost for Write-optimized data structure

Back

Quotient filters use less space than Bloom filters for all practical configurations



Bloom filter: ~*1.44 log*(*1/ε*) bits/element.

Quotient filter: ~*2.125* + *log*(*1/ε*) bits/element.

Cyber monitoring → real-time data analysis

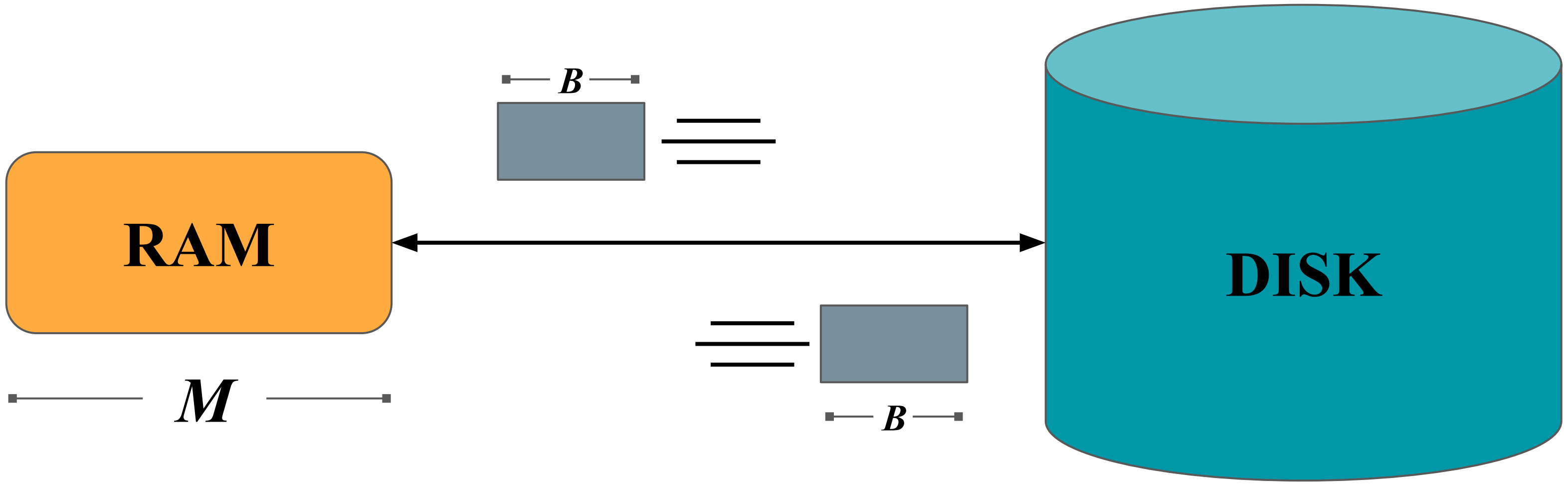
Defense systems for cyber security monitor ***high-speed*** streams for malicious traffic over ***large periods*** of time ➡

Malicious traffic forms a ***small portion*** of the stream ➡

Automated systems take defensive actions for ***every reported event*** ➡

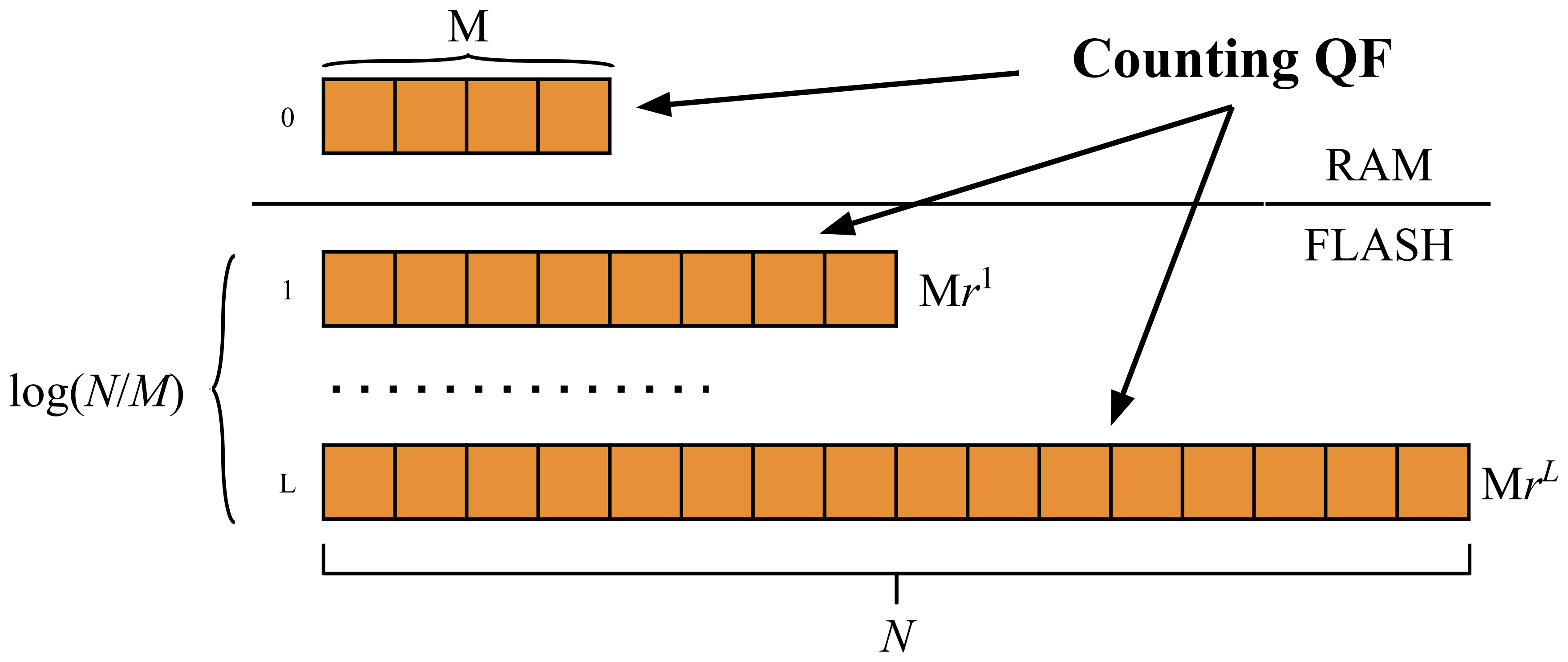
External memory model [Aggarwal+Vitter '08]

- **How computations work:**
  - Data is transferred in blocks between RAM and disk.
  - The number of block transfers dominate the running time.
- **Goal: Minimize number of block transfers**
  - Performance bounds are parameterized by block size B, memory size M, data size N.



Cascade filter: write-optimized quotient filter

[Bender et al. ‘12, Pandey et al. ‘17]

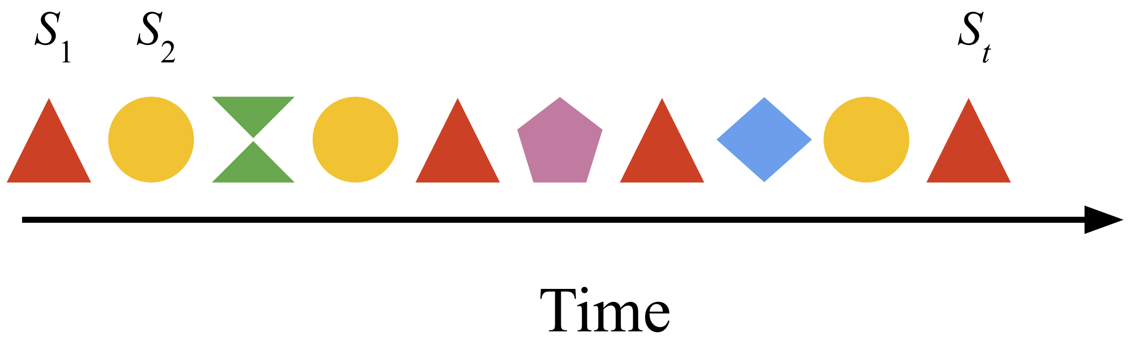


- The Cascade filter efficiently scales out-of-RAM
- It accelerates insertions at some cost to queries

More

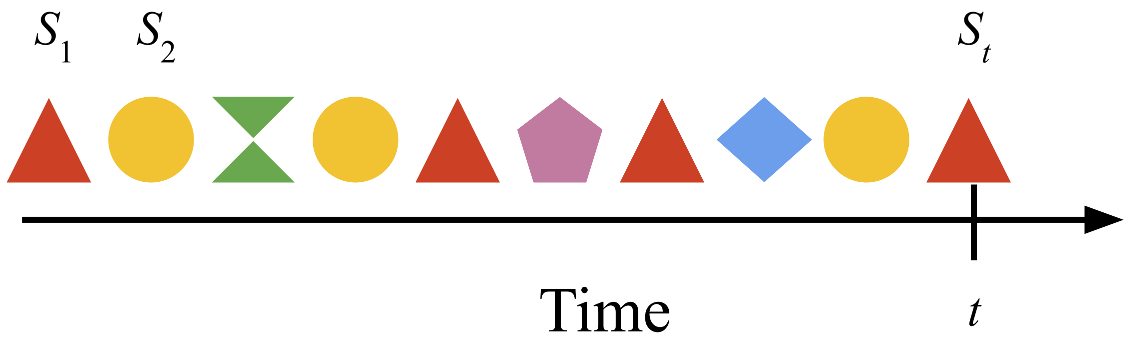
Timely event detection problem

- Stream of elements arrive over time



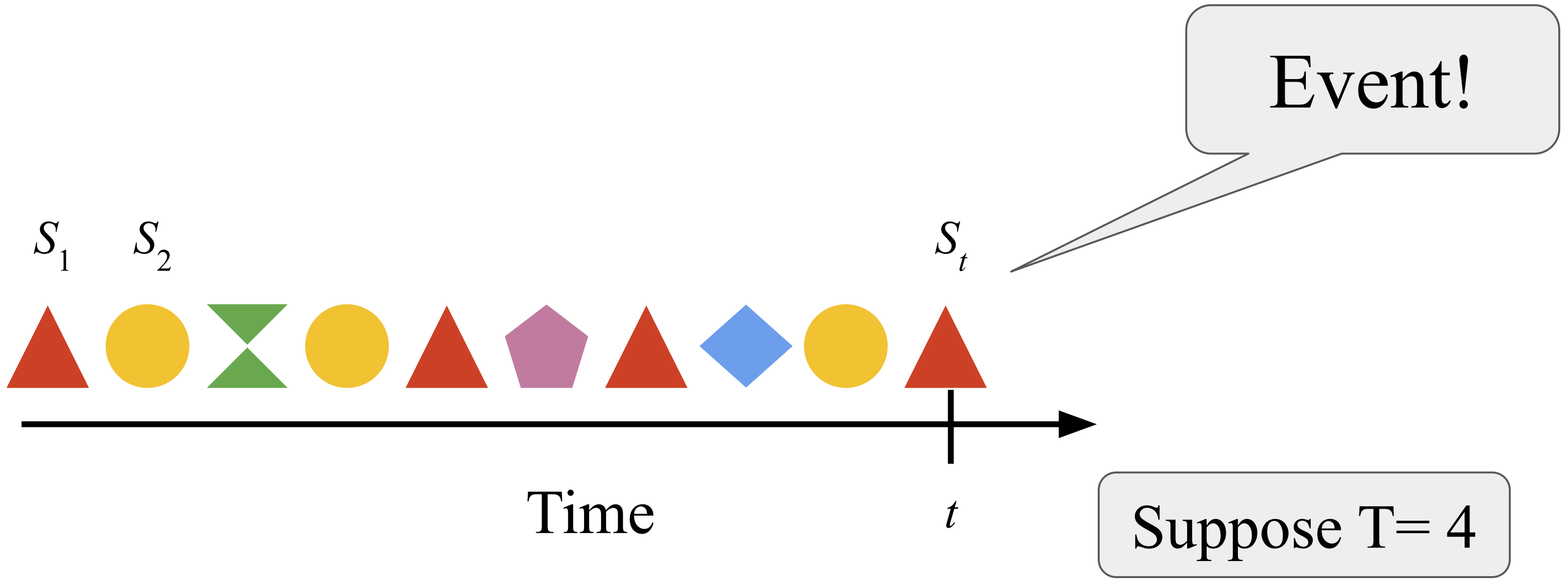
Timely event detection problem

- Stream of elements arrive over time
- An **event** occurs at time *t* if S_t occurs exactly *T* times in $(s_1, s_2 \ldots s_t)$



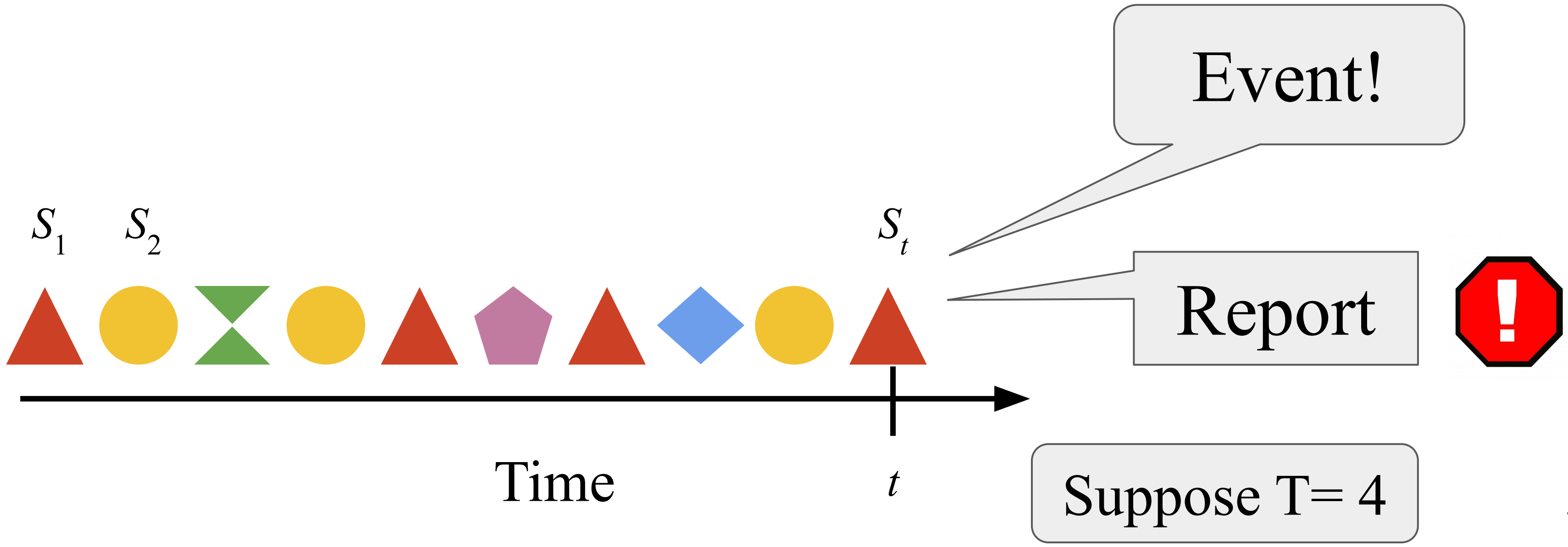
Timely event detection problem

- Stream of elements arrive over time
- An **event** occurs at time *t* if S_t occurs exactly T times in $(s_1, s_2 \ldots s_t)$

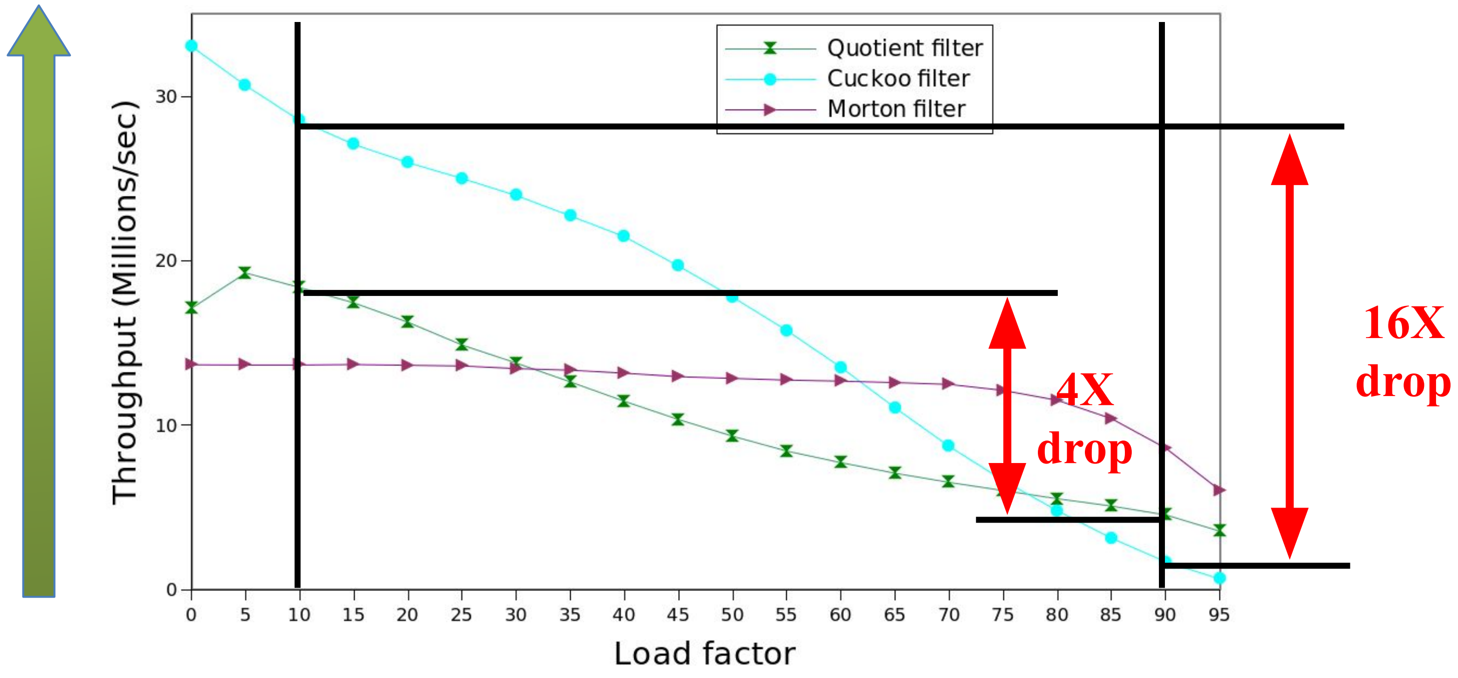


Timely event detection problem

- Stream of elements arrive over time
- An **event** occurs at time t if S_t occurs exactly T times in $(s_1, s_2 \ldots s_t)$
- In **timely event-detection problem (TED)**, we want to report all events shortly after they occur.



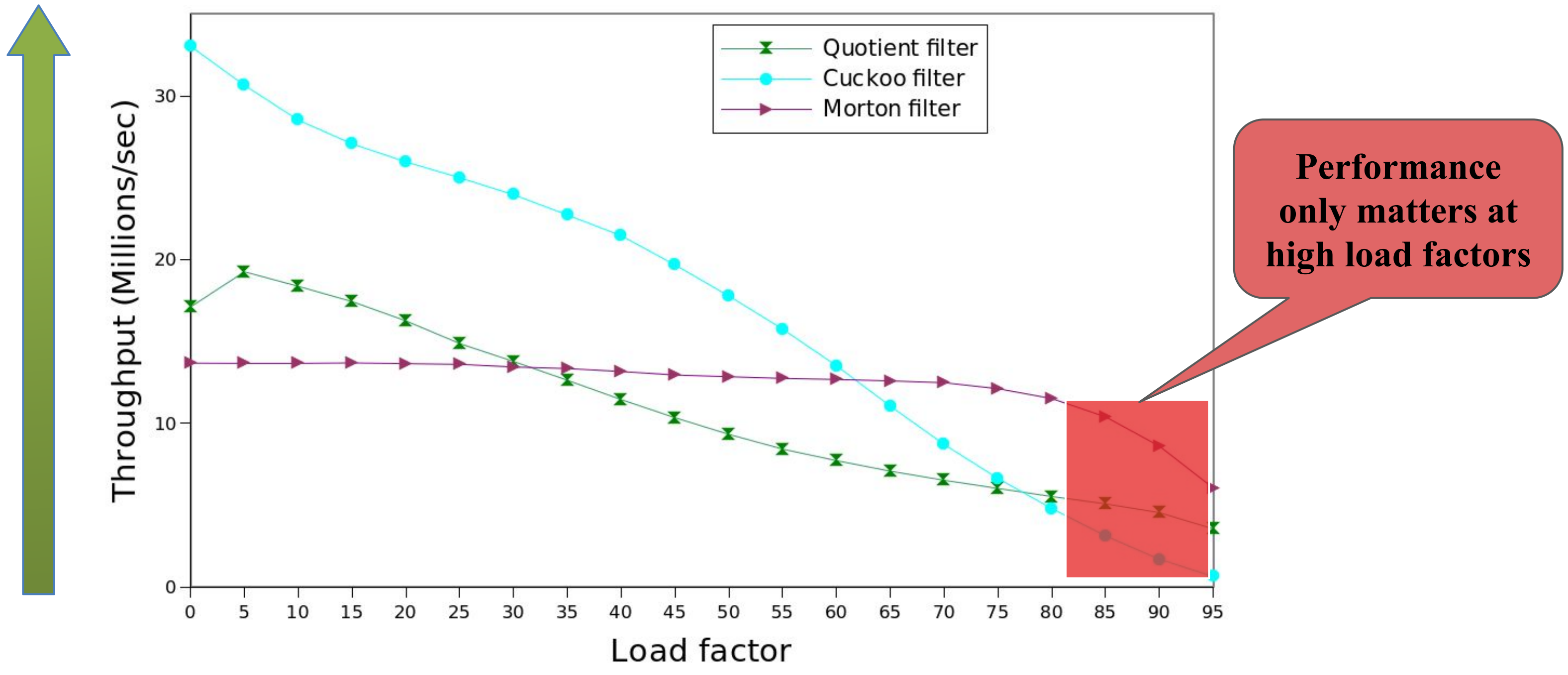
Trade-off: Insertion throughput degrades with load factor



Performance suffers due to high-overhead of ***collision resolution***

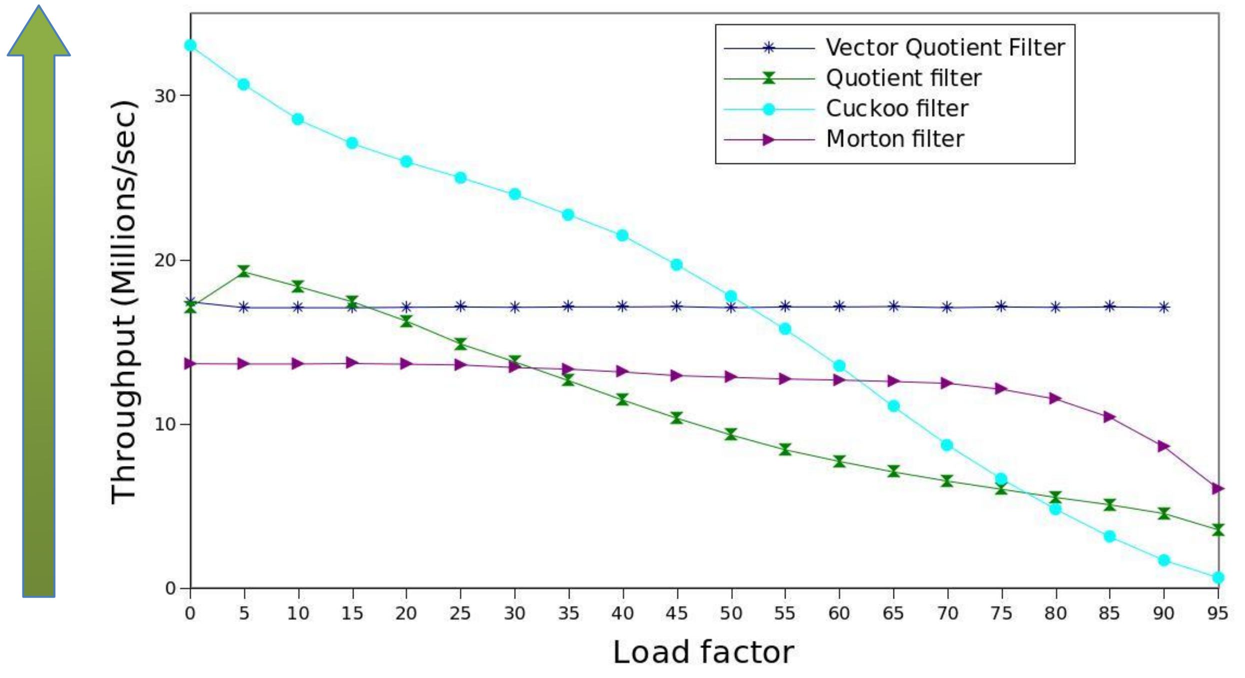
Trade-off: Insertion throughput degrades with load factor

Insertion throughput vs load factor of state-of-the-art filters



Many update-intensive applications (e.g., network caches, data analytics, etc.) maintain filters at high load factors

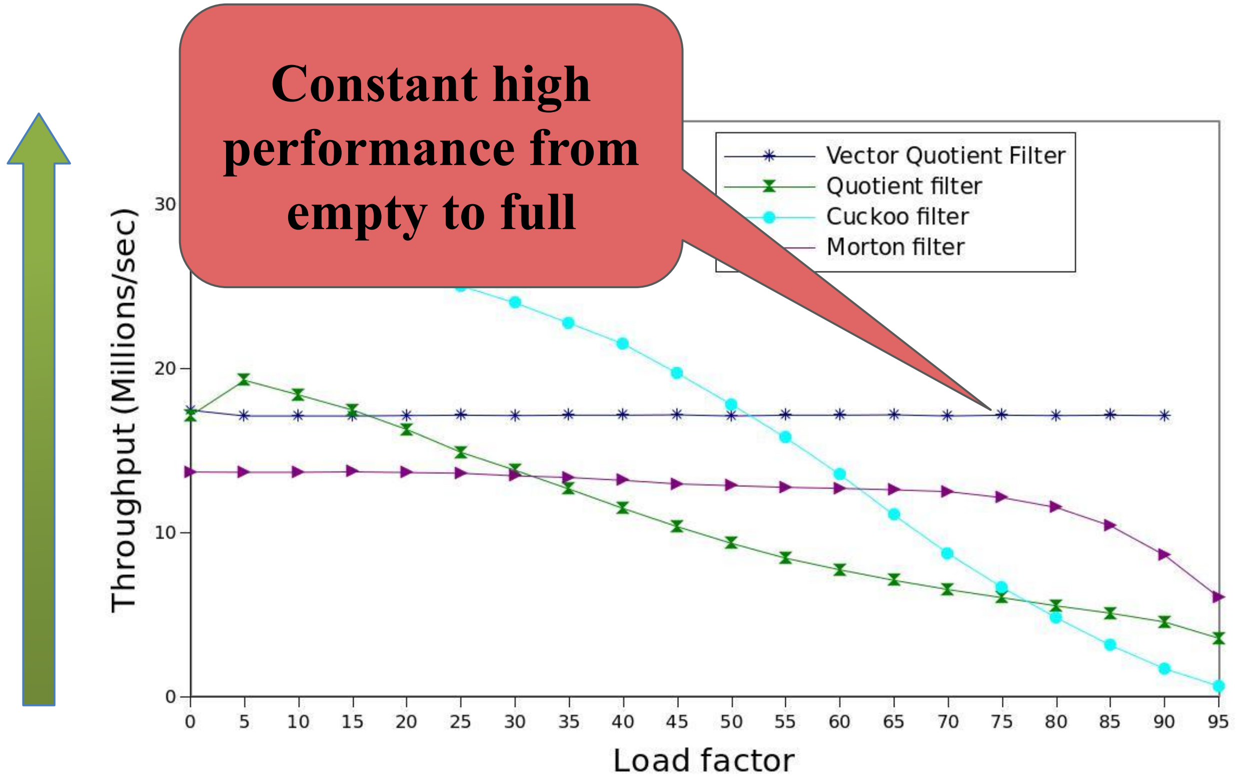
Combining techniques + new hardware



Combining hashing techniques (**Robin Hood + 2-choice hashing**)
Using ultra-wide vector operations (**AVX512-BW**)

Combining techniques + new hardware

Pandey et al. SIGMOD ‘21



Combining hashing techniques (**Robin Hood + 2-choice hashing**)

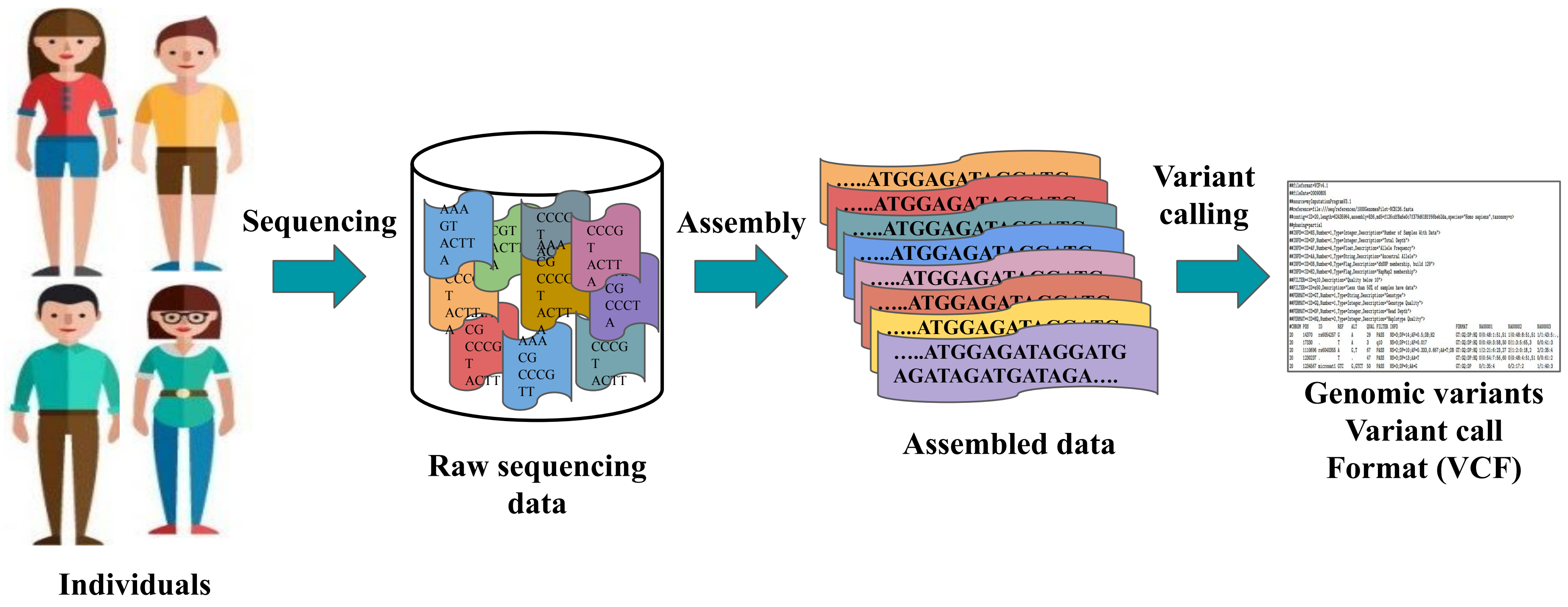
Using ultra-wide vector operations (**AVX512-BW**)

More

Future work: population-scale variant index

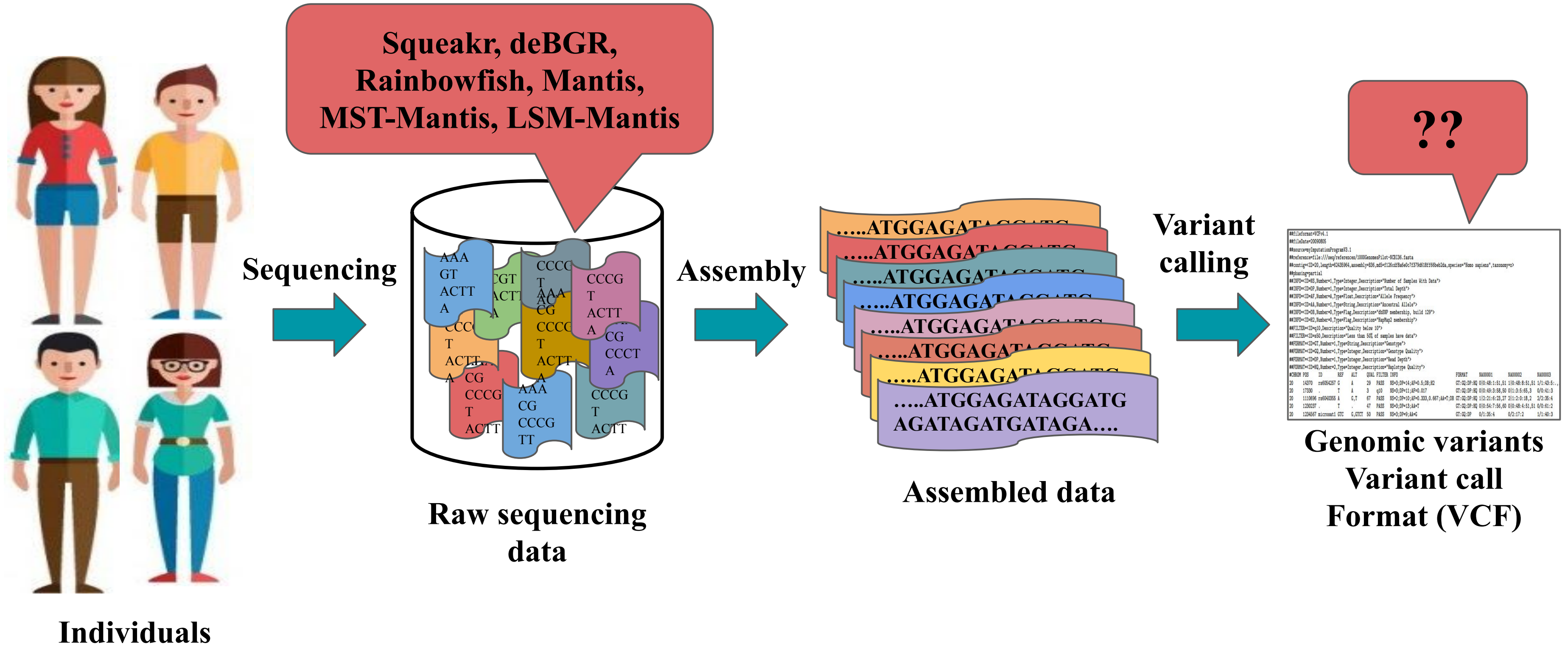
Goal: Build a ***population-scale*** index on variation data to enable downstream apps to gain ***quick insights into variants***

Country-scale sequencing efforts produce huge amounts of sequencing data



- 1000 Genomes project [https://www.internationalgenome.org/]
- The Cancer Genome Atlas (TCGA) [https://portal.gdc.cancer.gov/]
- Genotype-Tissue Expression (GTEx) [https://gtexportal.org/home/]

Country-scale sequencing efforts produce huge amounts of sequencing data

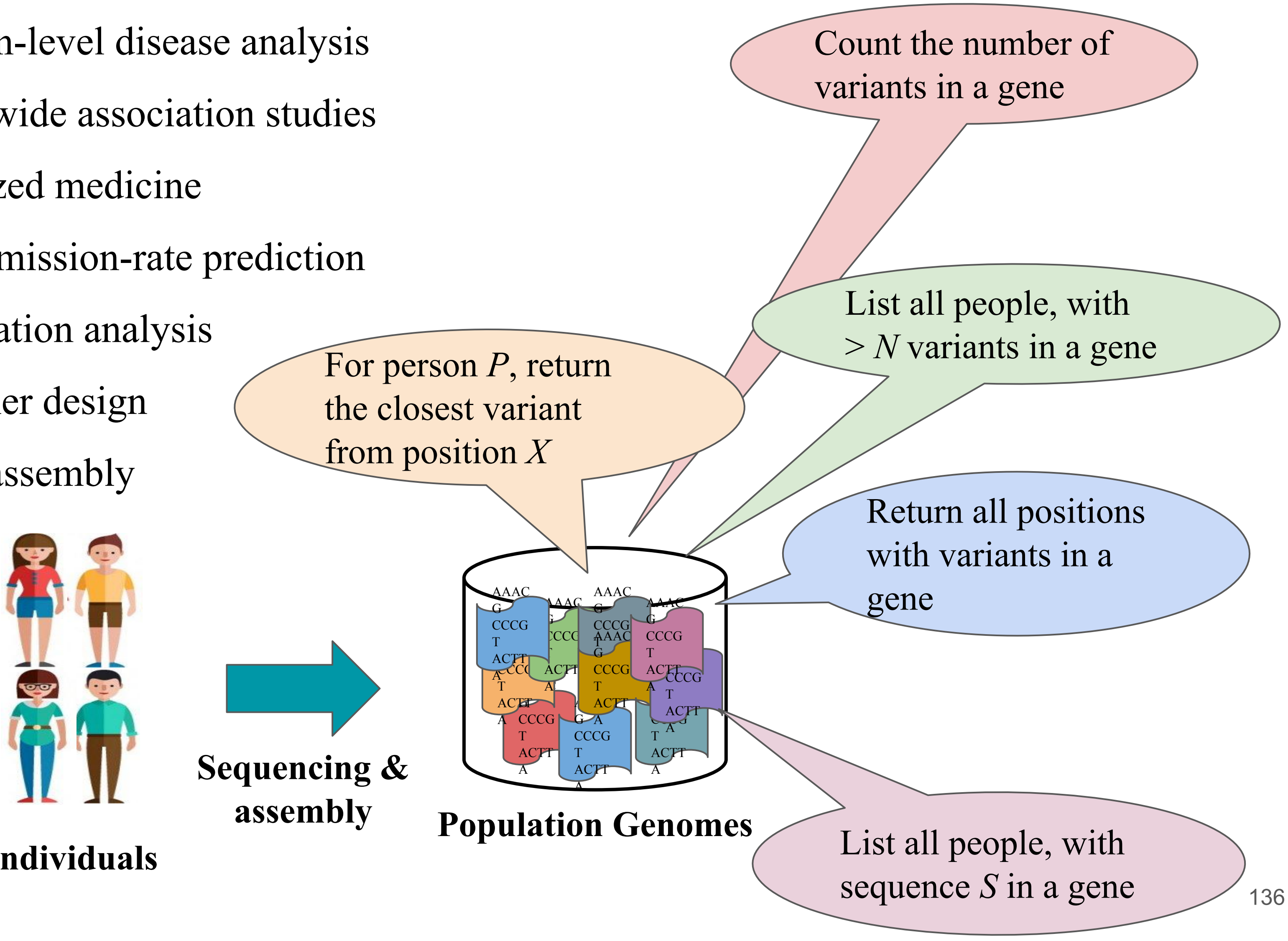


Squeakr, deBGR, Rainbowfish, Mantis, MST-Mantis, LSM-Mantis
Sequencing
Assembly
Variant calling
??
Individuals
Raw sequencing data
Assembled data
Genomic variants Variant call Format (VCF)
…..ATGGAGATAGGATG AGATAGATGATAGA….

- 1000 Genomes project [https://www.internationalgenome.org/]
- The Cancer Genome Atlas (TCGA) [https://portal.gdc.cancer.gov/]
- Genotype-Tissue Expression (GTEx) [https://gtexportal.org/home/]

Variation data analysis can improve downstream applications

- Population-level disease analysis
- Genome-wide association studies
- Personalized medicine
- Cancer remission-rate prediction
- Colocalization analysis
- PCR primer design
- Genome assembly


Count the number of variants in a gene
List all people, with > *N* variants in a gene
For person *P*, return the closest variant from position *X*
Return all positions with variants in a gene
List all people, with sequence *S* in a gene
Individuals
Sequencing & assembly
Population Genomes
AAAC G CCCG T ACTT A

Indexing in multiple coordinates is challenging

Reference-only indexes map positions only in the reference coordinate system

$$f(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j$$

Pan-genome analysis involves queries based on sample coordinate systems

$$\text{Num Samples} \begin{cases} f_1(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \\ \vdots \\ f_s(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j \end{cases}$$

Maintaining thousands of mappings ***increases*** computational ***complexity*** and ***memory footprint***

Limits scalability to population-scale data

Indexing in multiple coordinates is challenging

Reference-only indexes map positions only in the reference coordinate system

$$f(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j$$

Pan-genome analysis involves queries based on sample coordinate systems

Existing systems don't support multiple coordinate systems. The ones that do, don't *scale* beyond a few thousand samples.

Nu
Sam

$$f_s(p_i, p_j) \rightarrow (v_i \ldots v_n), \text{ where } p_i \leq p_j$$

Maintaining thousands of mappings ***increases*** computational ***complexity*** and ***memory footprint***

Limits scalability to population-scale data

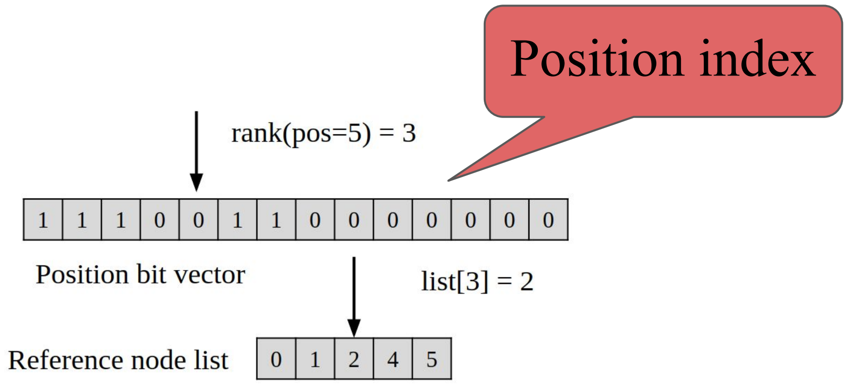
An inverted index on the pan-genome graph

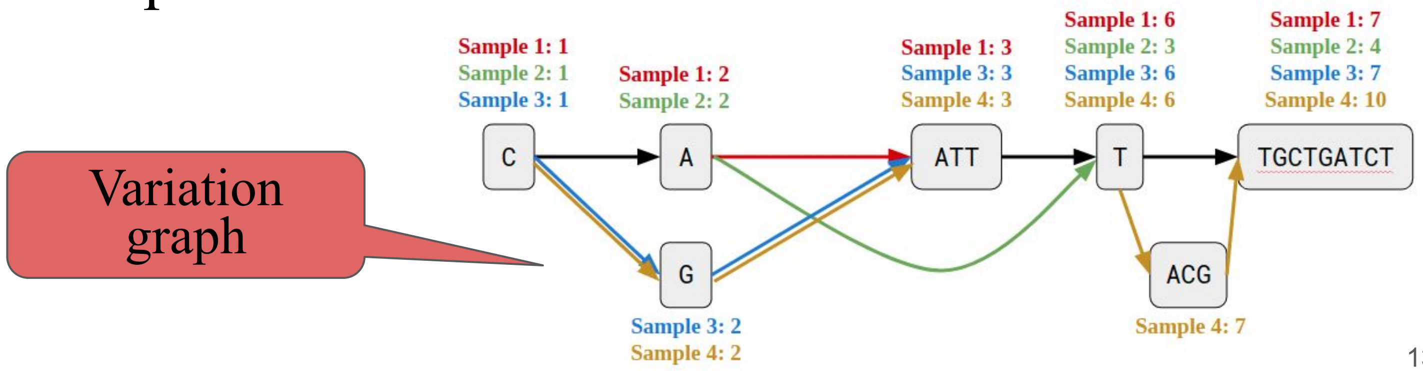
Collaborators: Yinjie Gao, Carl Kingsford

- Partition the variation graph based on coordinate ranges
- Store partitions on disk

Queries often require loading 1-2 partitions

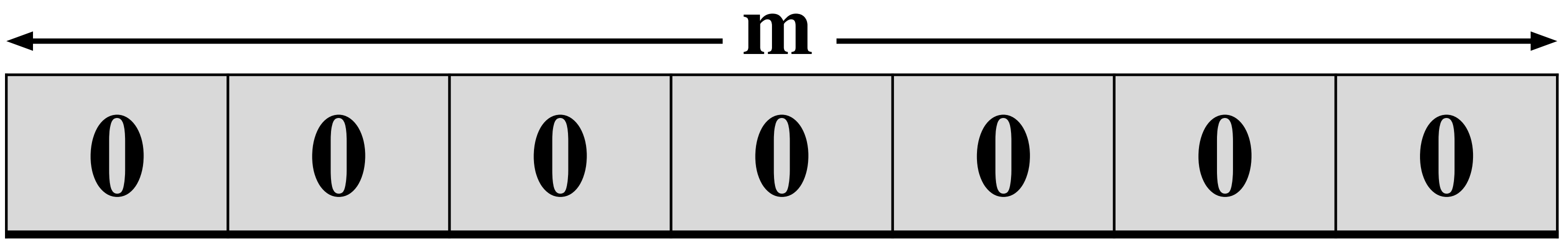
- Succinct index for reference coordinate system
- Local-graph exploration to map position from reference to sample coordinate

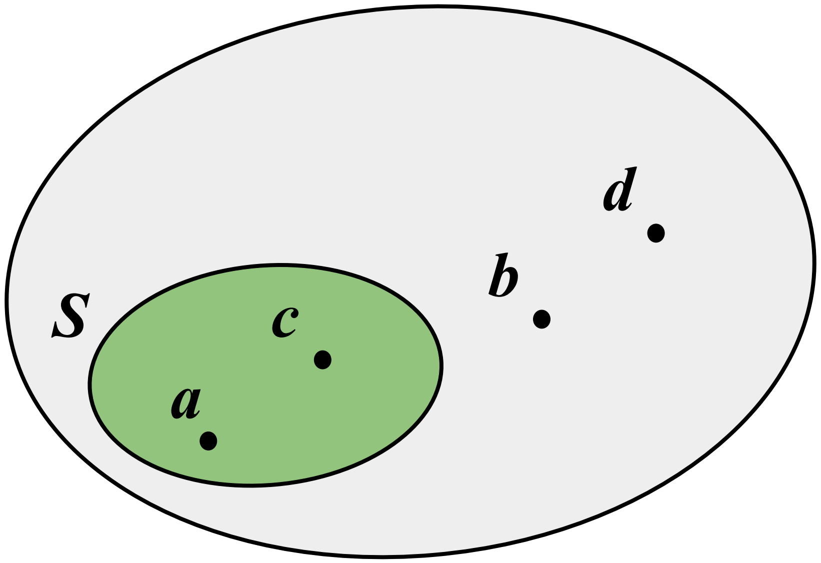




Classic filter: The Bloom filter [Bloom '70]

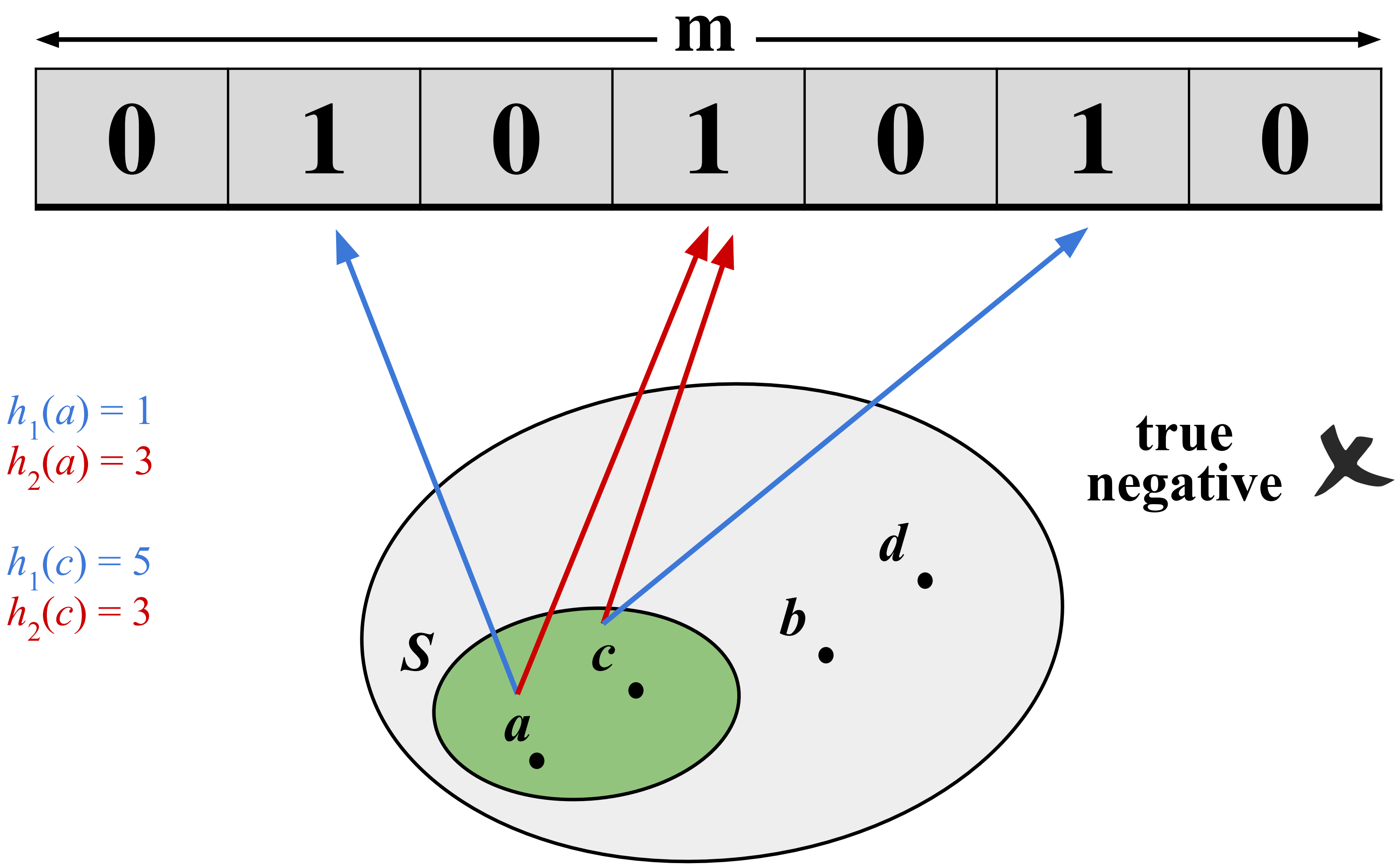
Bloom filter: a bit array + *k* hash functions





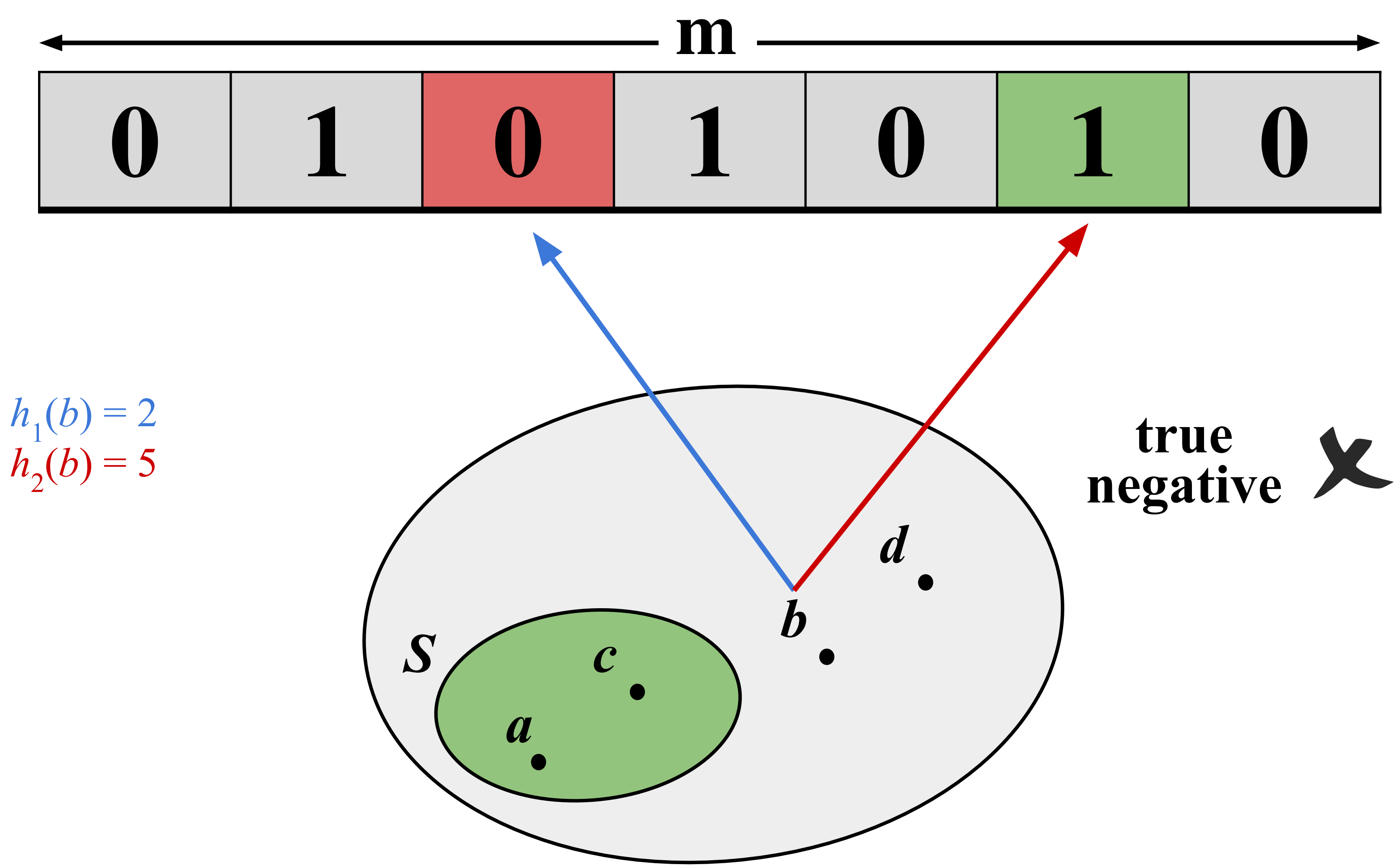
Classic filter: The Bloom filter [Bloom '70]

Bloom filter: a bit array + k hash functions (here $k = 2$)



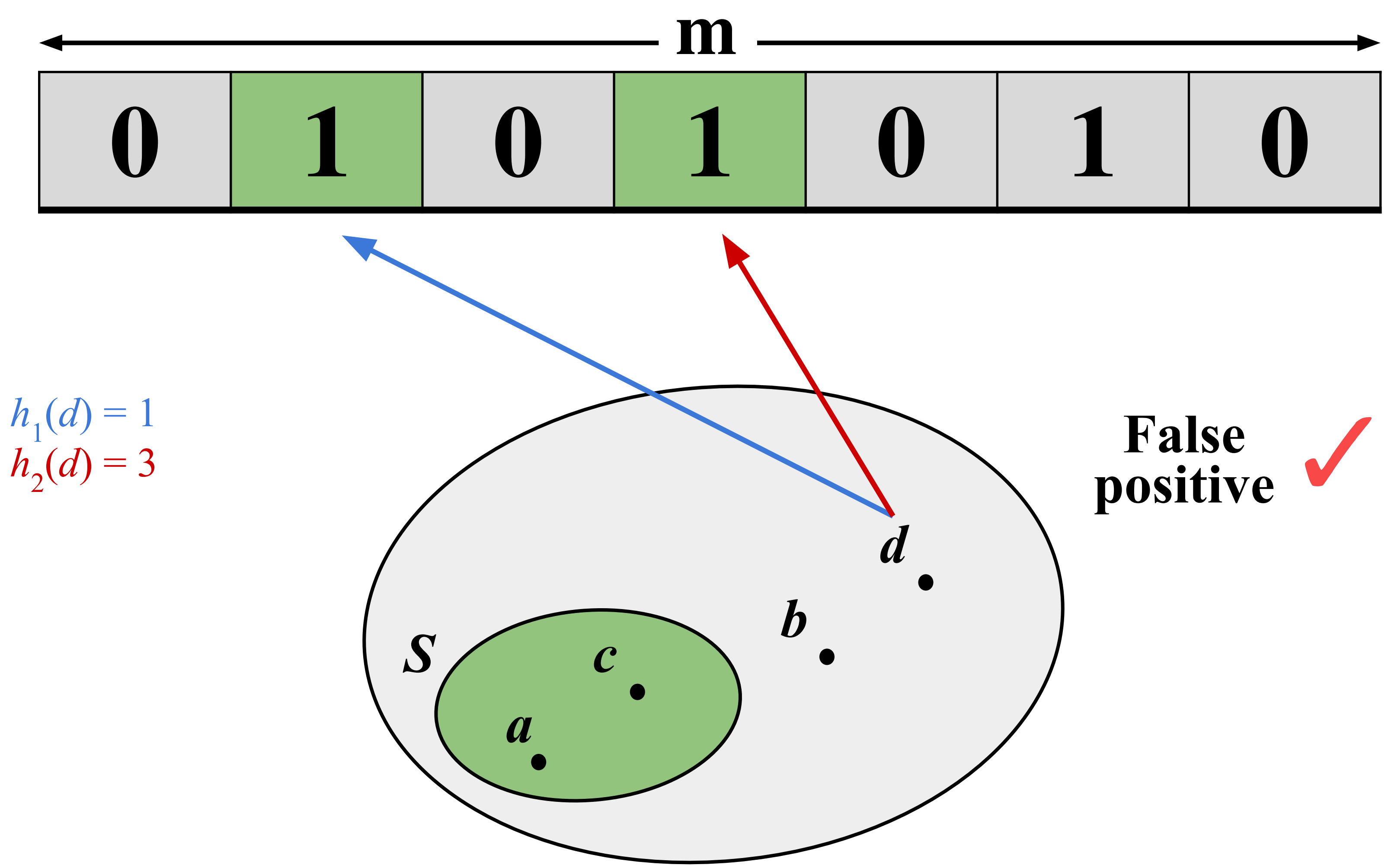
Classic filter: The Bloom filter [Bloom '70]

Bloom filter: a bit array + *k* hash functions (here *k*=2)



Classic filters: The Bloom filter [Bloom '70]

Bloom filter: a bit array + *k* hash functions (here *k*=2)



Bloom filters are ubiquitous (> 4300 citations)

Streaming applications

Networking

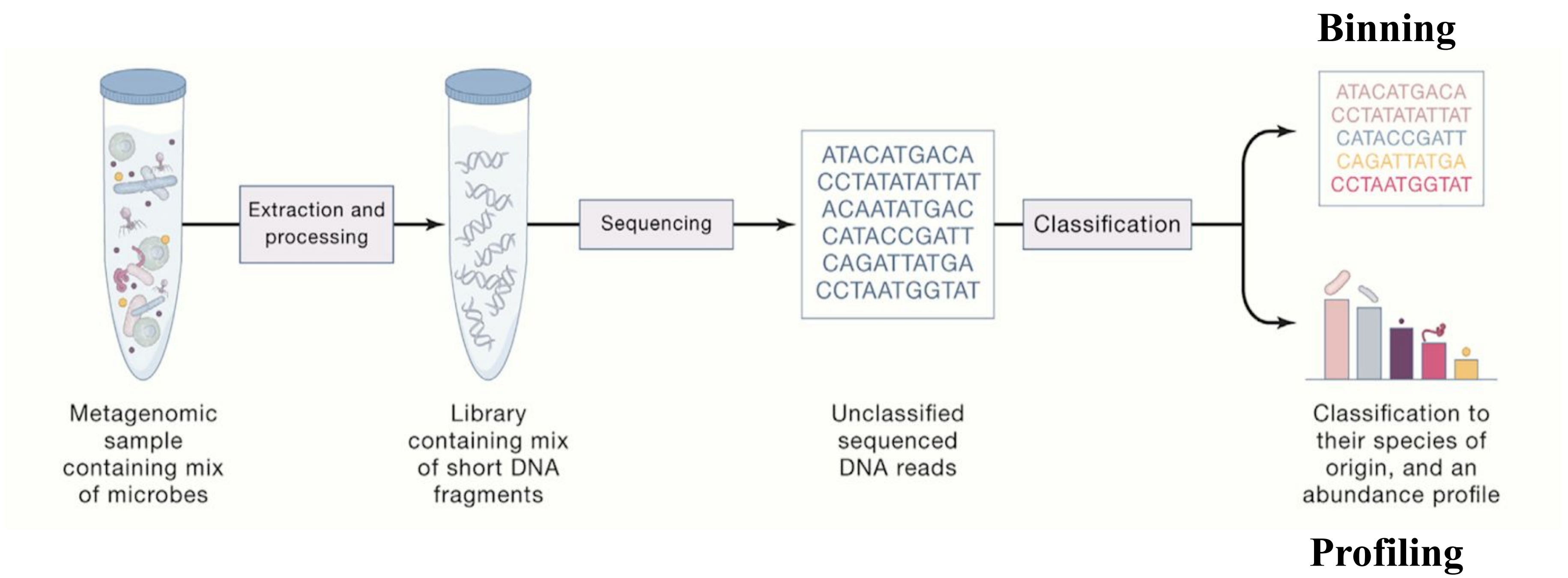
Databases

Computational biology

Storage systems

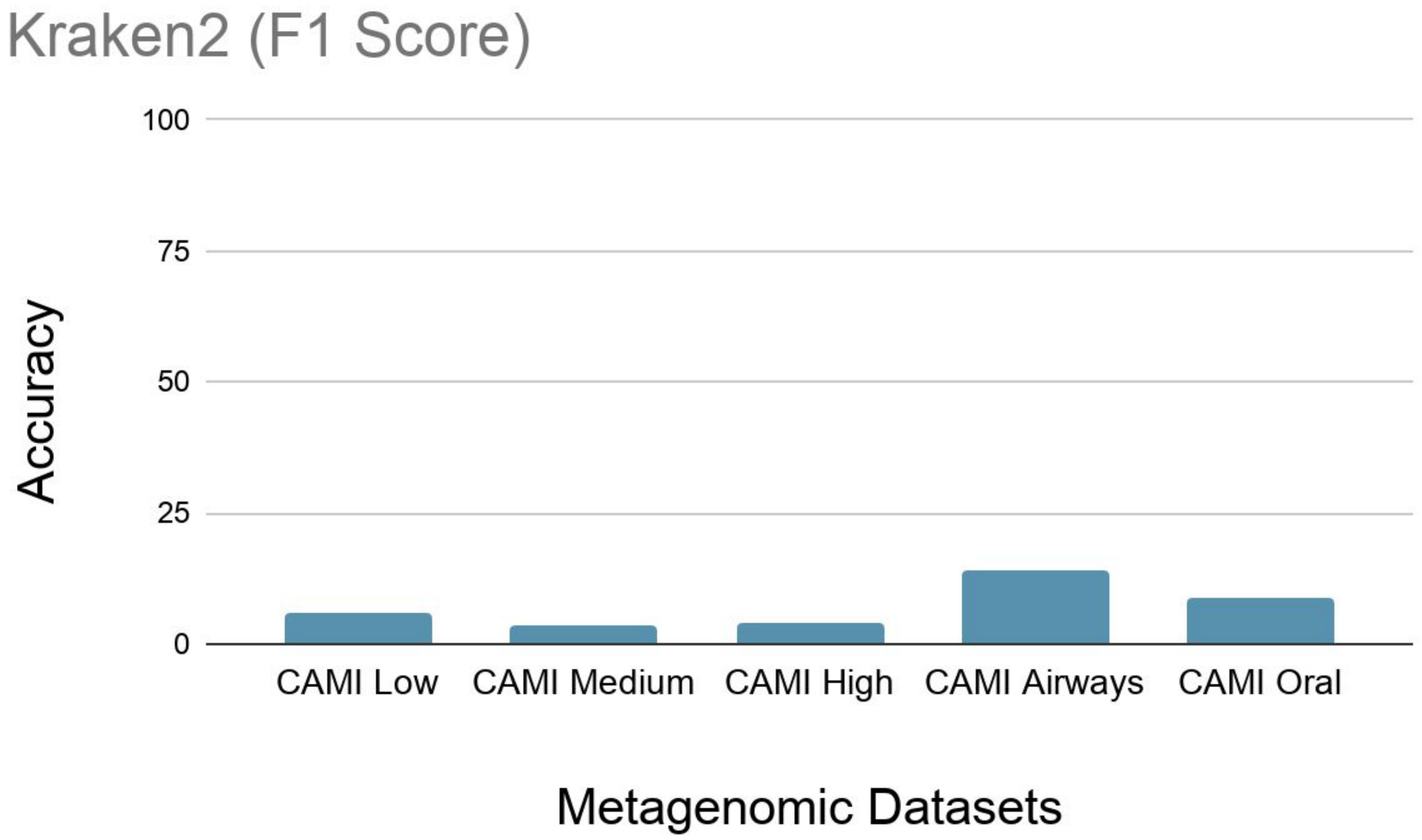
Metagenomic classification pipeline

[Ye et al. 2019]


Binning
Extraction and processing
Sequencing
ATACATGACA
CCTATATATTAT
ACAATATGAC
CATACCGATT
CAGATTATGA
CCTAATGGTAT
Classification
ATACATGACA
CCTATATATTAT
CATACCGATT
CAGATTATGA
CCTAATGGTAT
Metagenomic sample containing mix of microbes
Library containing mix of short DNA fragments
Unclassified sequenced DNA reads
Classification to their species of origin, and an abundance profile
Profiling

Classification is the ***critical first step*** in many metagenomic analysis pipeline

Existing indexing techniques offer low accuracy



Indexing-based classification is done based ***only on the contents on the input sequences***