RUBIK: Efficient Threshold Queries on Massive Time Series

Eleni Tzirita Zacharatou[‡]

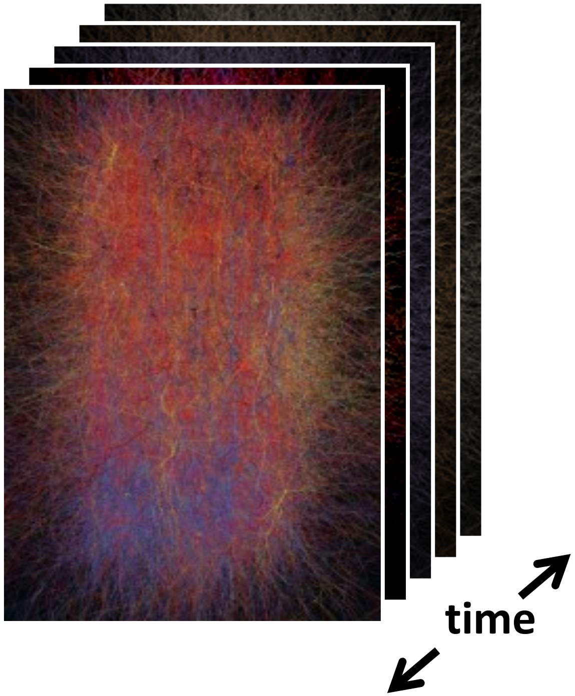
Thomas Heinis[*] *Farhan Tauheed*[§] *Anastasia Ailamaki*[‡]

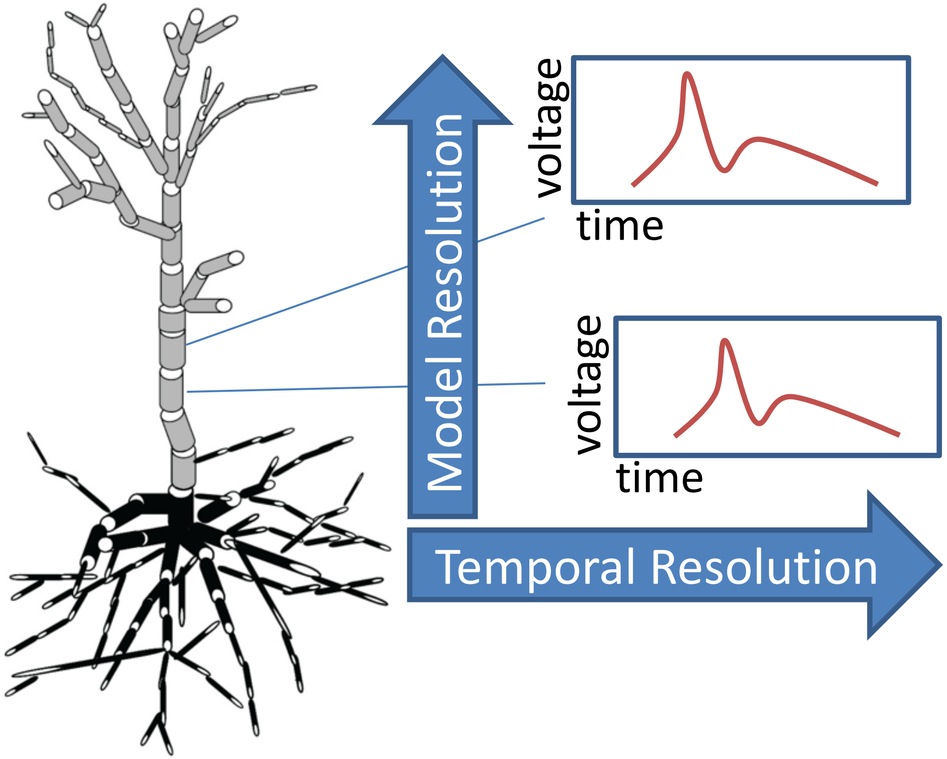
[*]Imperial College London

[§]Oracle Labs, Zurich

[‡]École Polytechnique
Fédérale de Lausanne

Scaling up Brain Simulations





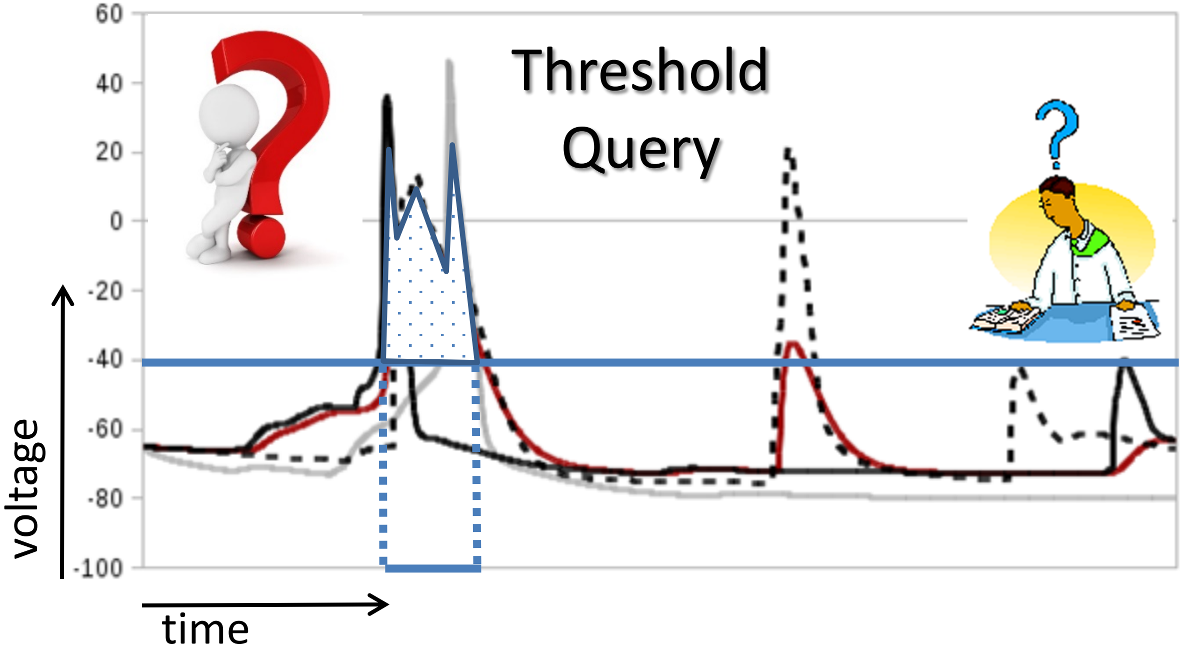
3D Neuron Model

Time Series Analysis: key to neuroscientific discovery

Neuron firing: which and when

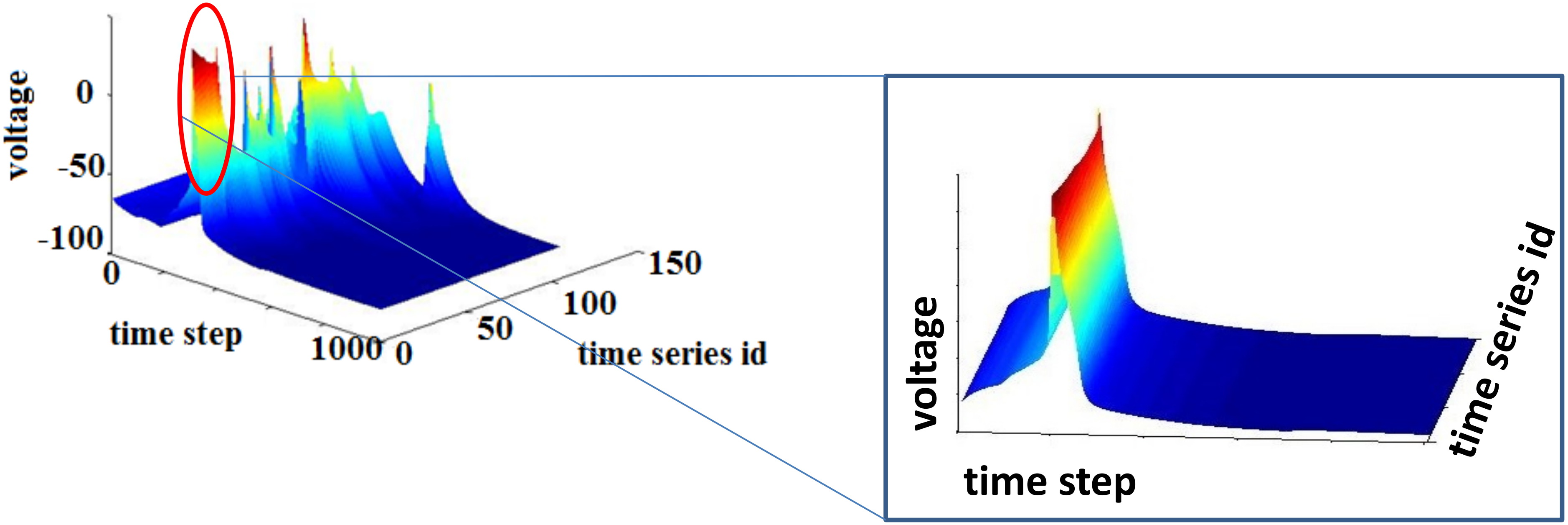
- Exploration
- Hypothesis Testing

- Identify subsets of interest:
  time series where voltage > -40
  and time step ∈ [300,400]



Threshold queries fuel efficient data analysis

Time Series Correlation…



Trends	Correlation	Opportunity to scale with
Increased simulation duration	Across time	increase in temporal resolution
Increasingly detailed models	Across time series	increase in spatial resolution

…enables efficient time series-specific compression

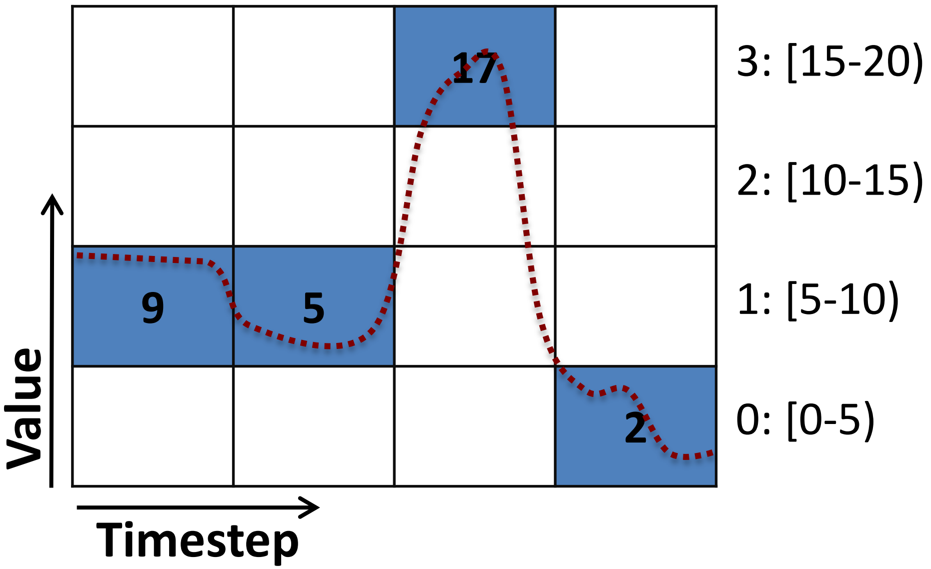
Time Series Data Discretization

Binning:

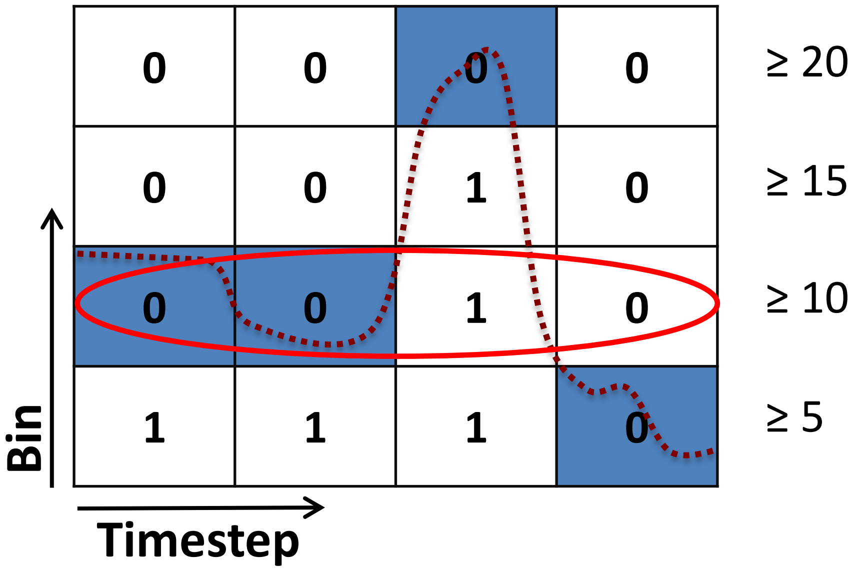
Partition the values into bins

Range encoding:

Set bin to ‘1’ if condition satisfied, ‘0’ otherwise



		17		3: [15-20)
				2: [10-15)
9	5			1: [5-10)
			2	0: [0-5)



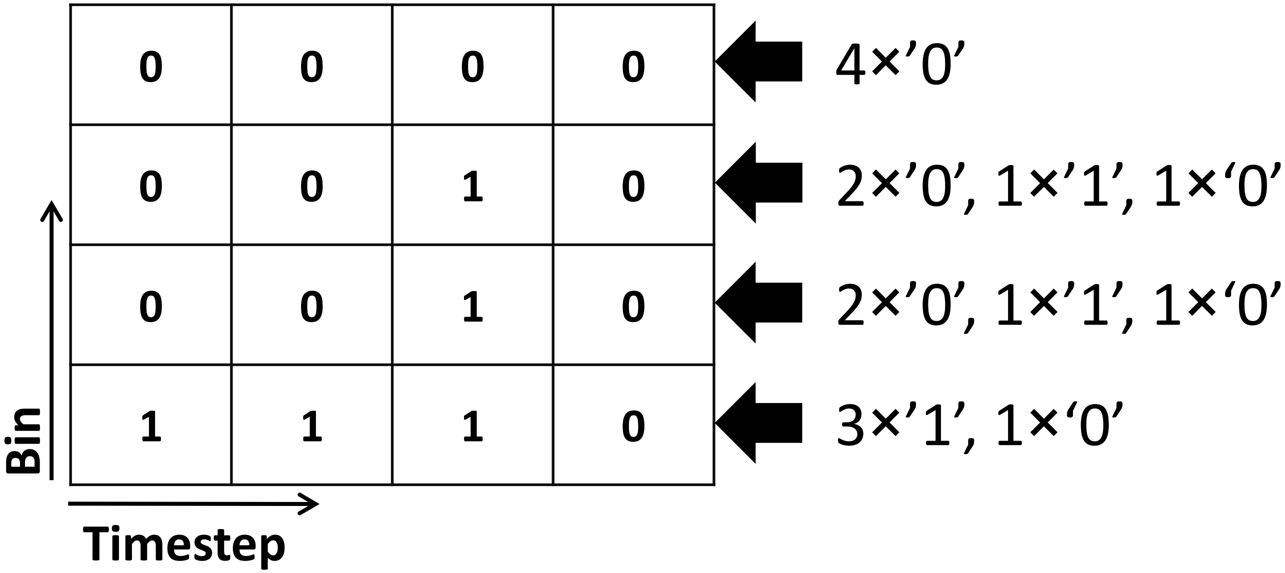
0	0	0	0	≥ 20
0	0	1	0	≥ 15
0	0	1	0	≥ 10
1	1	1	0	≥ 5

Increased similarity across time series

Precomputed answers stored as a bitmap

Bitmap Compression Today

- Run-Length-Encoding compresses each bitvector
  - Word-Aligned Hybrid Code (WAH) [SSDBM ’02]

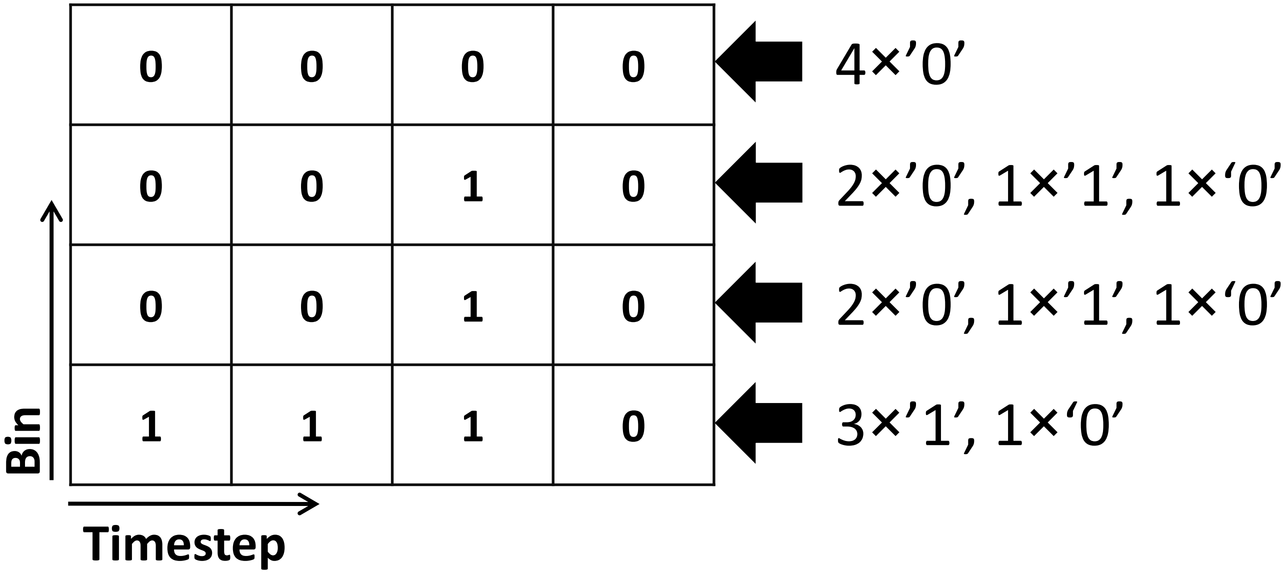


0	0	0	0	4×’0’
0	0	1	0	2×’0’, 1×’1’, 1×‘0’
0	0	1	0	2×’0’, 1×’1’, 1×‘0’
1	1	1	0	3×’1’, 1×‘0’

- Compression prevents direct access
  - Timesteps don’t correspond to bit positions

Bitmap Compression Today

- Run-Length-Encoding compresses each bitvector
  - Word-Aligned Hybrid Code (WAH) [SSDBM ’02]



0	0	0	0
0	0	1	0
0	0	1	0
1	1	1	0

4×’0’

2×’0’, 1×’1’, 1×‘0’

2×’0’, 1×’1’, 1×‘0’

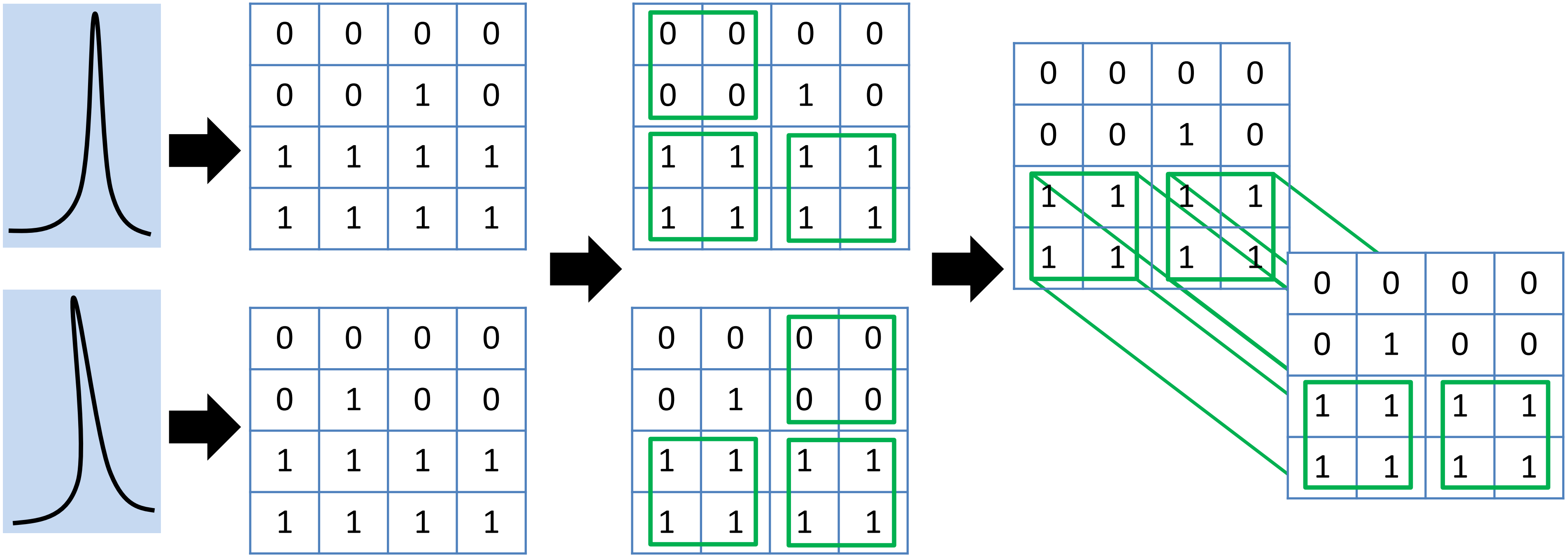
3×’1’, 1×‘0’

- Compression prevents direct access

Values filtered independently of timesteps

Similarities across time series are not exploited

Our Approach: RUBIK



0	0	0	0
0	0	1	0
1	1	1	1
1	1	1	1

0	0	0	0
0	1	0	0
1	1	1	1
1	1	1	1

0	0	0	0
0	0	1	0
1	1	1	1
1	1	1	1

0	0	0	0
0	1	0	0
1	1	1	1
1	1	1	1

0	0	0	0
0	0	1	0
1	1	1	1
1	1	1	1

0	0	0	0
0	1	0	0
1	1	1	1
1	1	1	1

Bitmap index creation

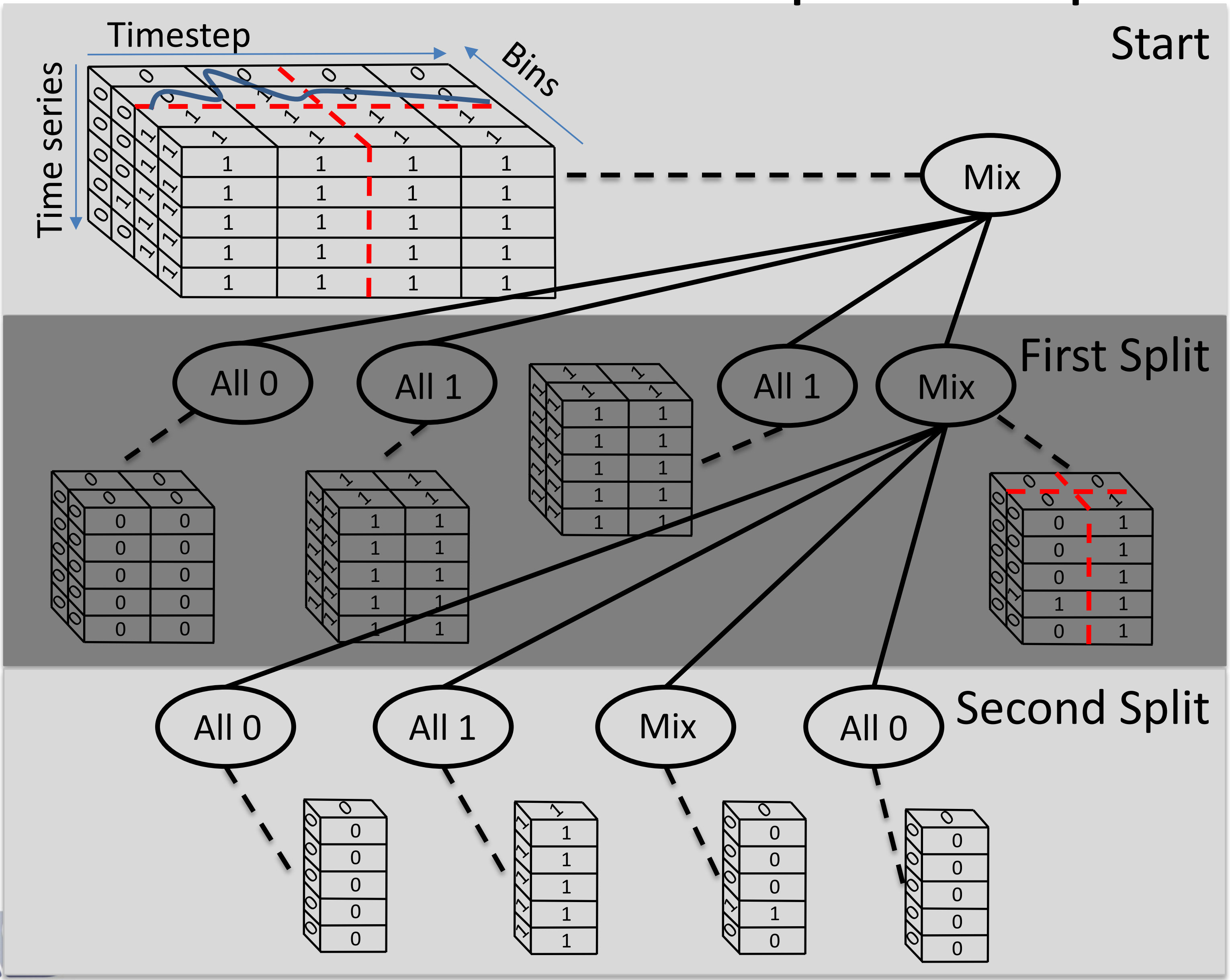
Quadtree-based bitmap decomposition

Bitmap stacking

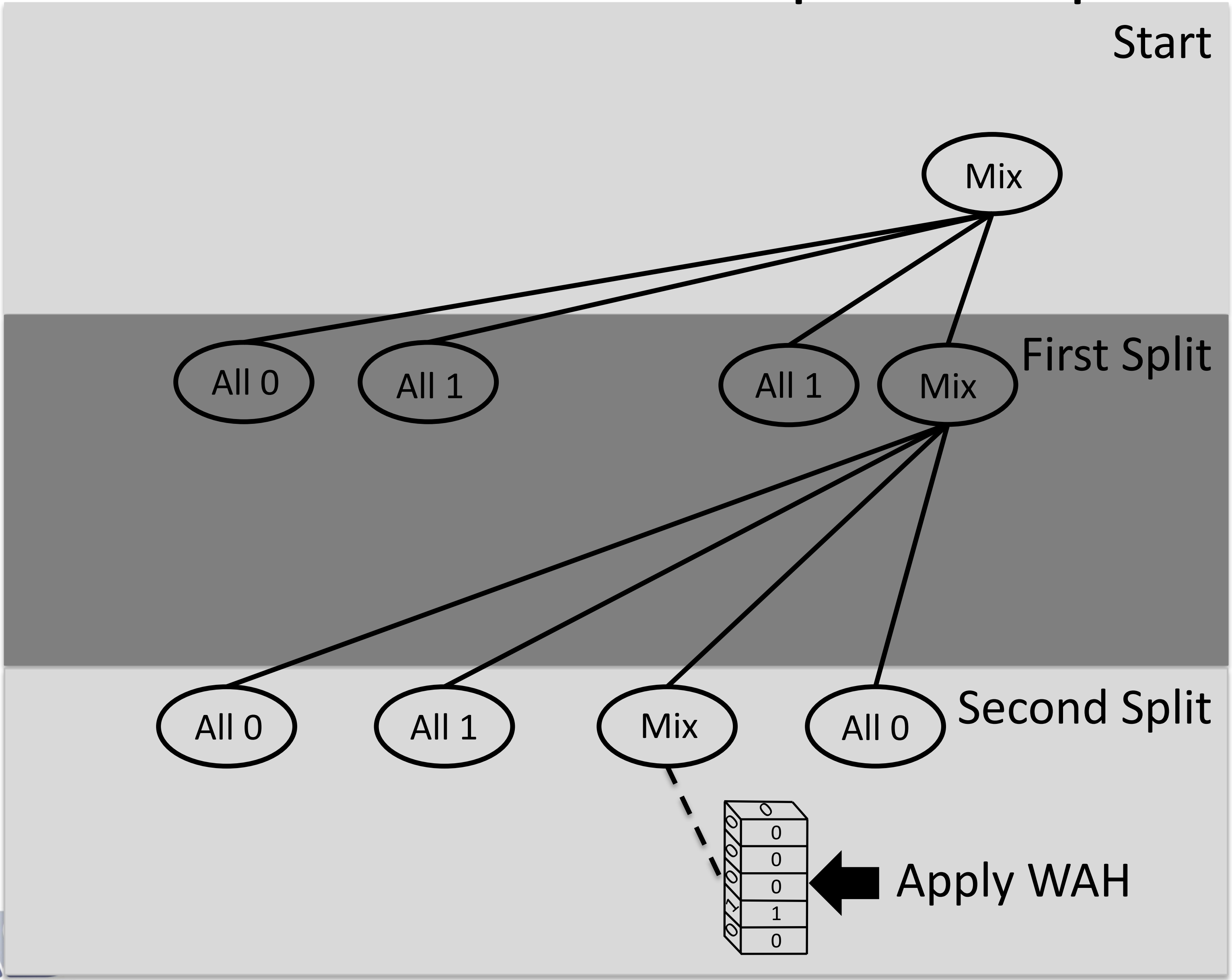
Access specific timesteps

Exploit similarities

Quadtree-based 3D Bitmap Decomposition



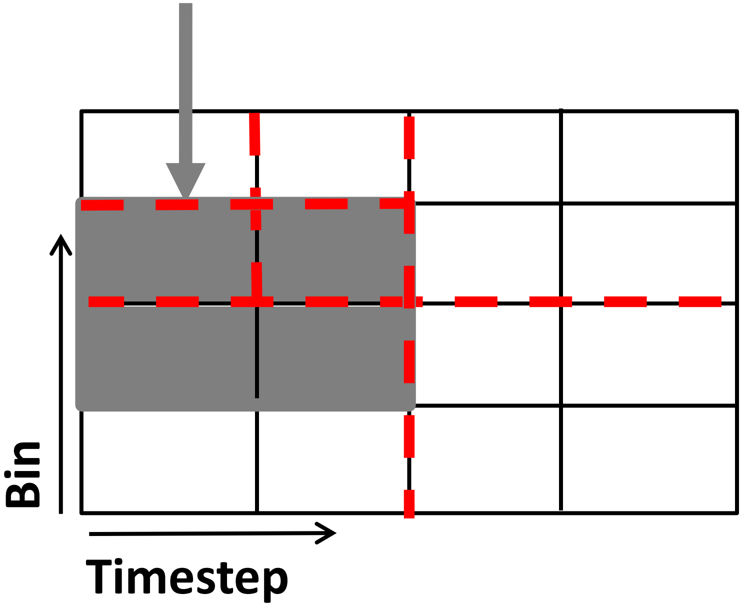
Quadtree-based 3D Bitmap Decomposition

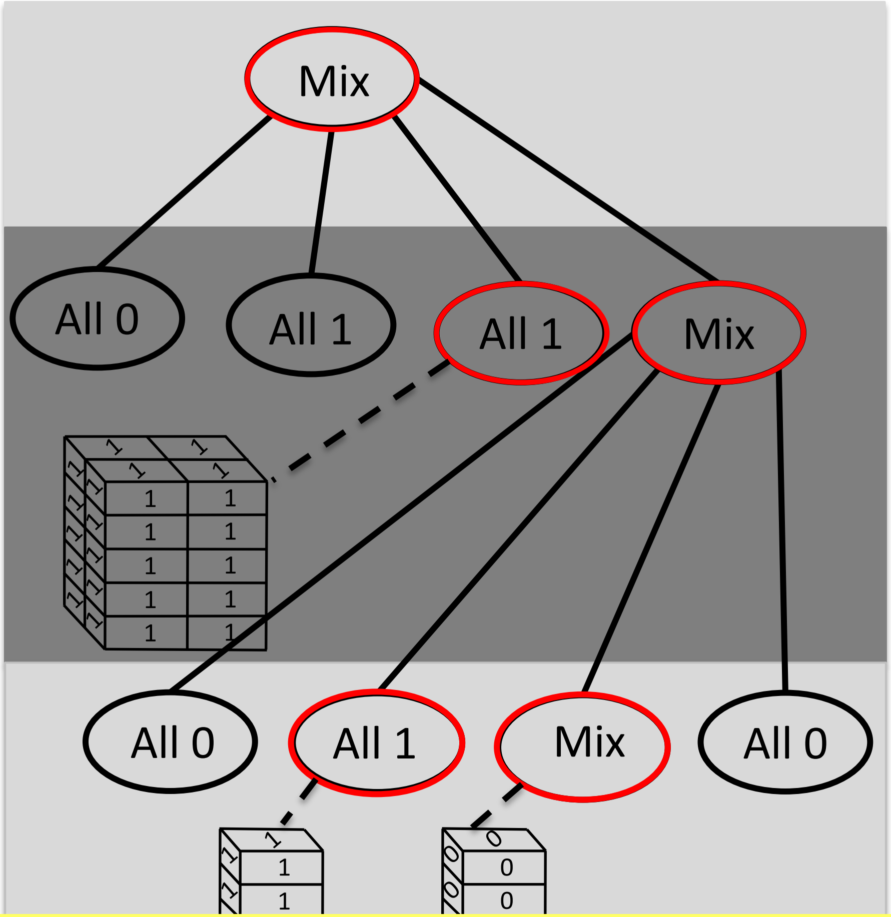


Query Execution

Query:

voltage > 11 in time steps 1 and 2



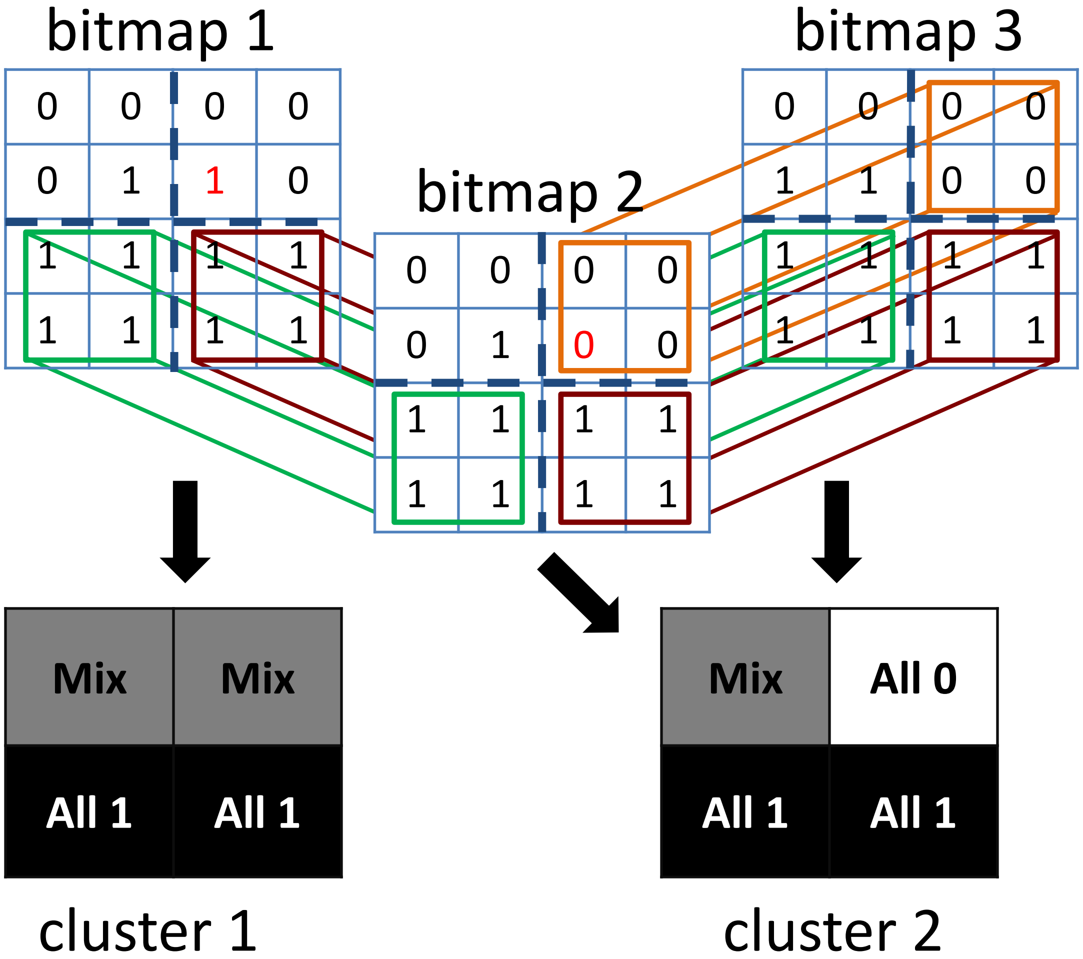


Transformation into a 2D bitmap problem

One tree traversal to retrieve multiple bitmaps

Stacking Time Series Bitmaps

Goal: Maximize size and number of common squares



bitmap 1

0	0	0	0
0	1	1	0
1	1	1	1
1	1	1	1

bitmap 2

0	0	0	0
0	1	0	0
1	1	1	1
1	1	1	1

bitmap 3

0	0	0	0
1	1	0	0
1	1	1	1
1	1	1	1

Mix	**Mix**
All 1	**All 1**

cluster 1

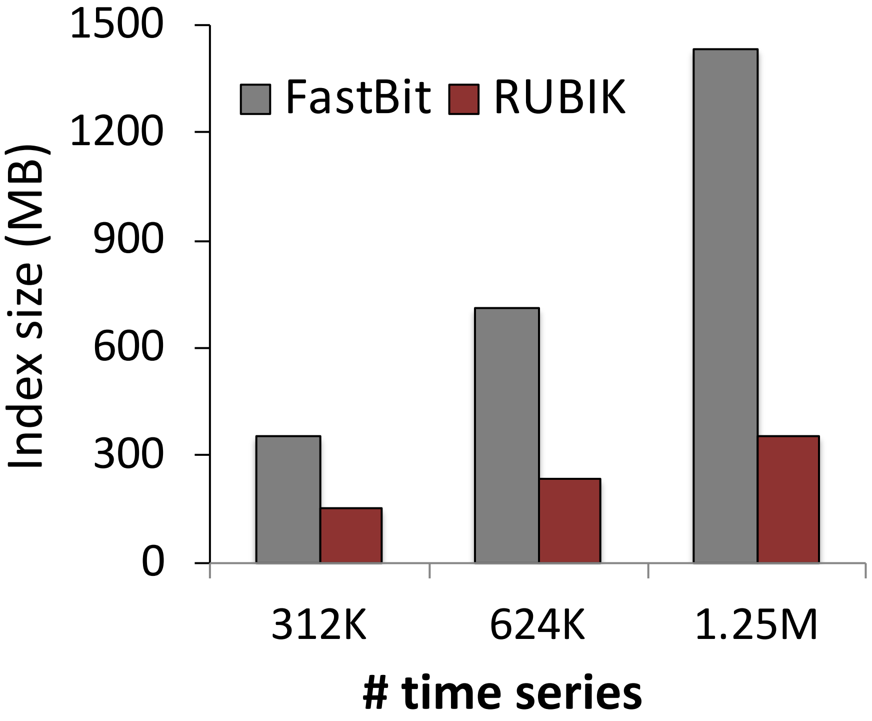
Mix	**All 0**
All 1	**All 1**

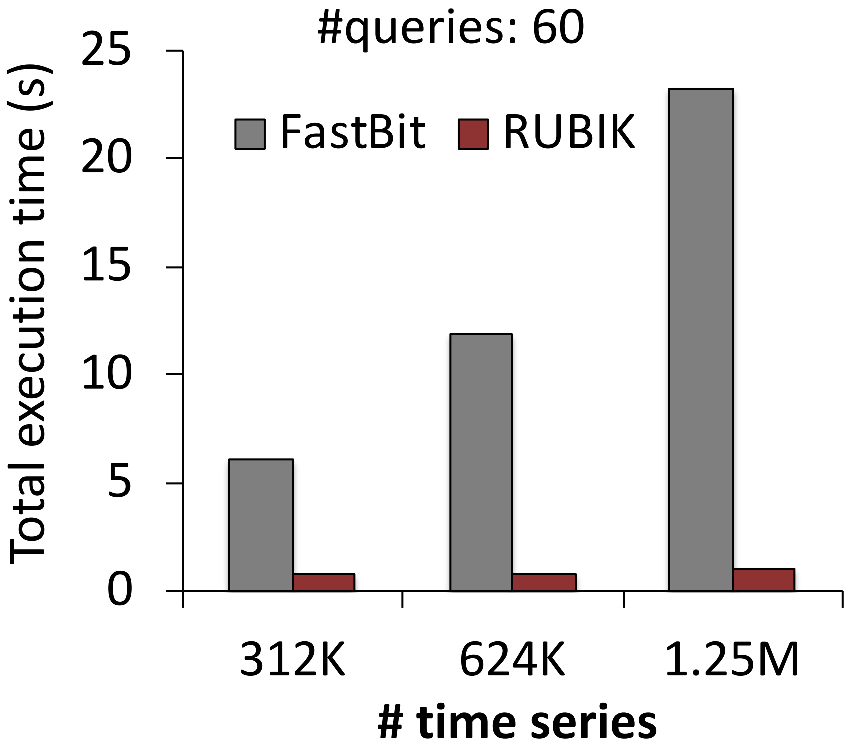
cluster 2

⇒ Maximize compression across time series

Scaling with Data Volume

In-memory indexes: FastBit (WAH-compressed bitmap index) and RUBIK
Configuration: 128 bins, **Hardware:** AMD Opteron CPU @ 2.7GHz, 32GB RAM
Time series data: 1000 time steps, 1.2GB – 4.8GB





RUBIK index size scales sublinearly

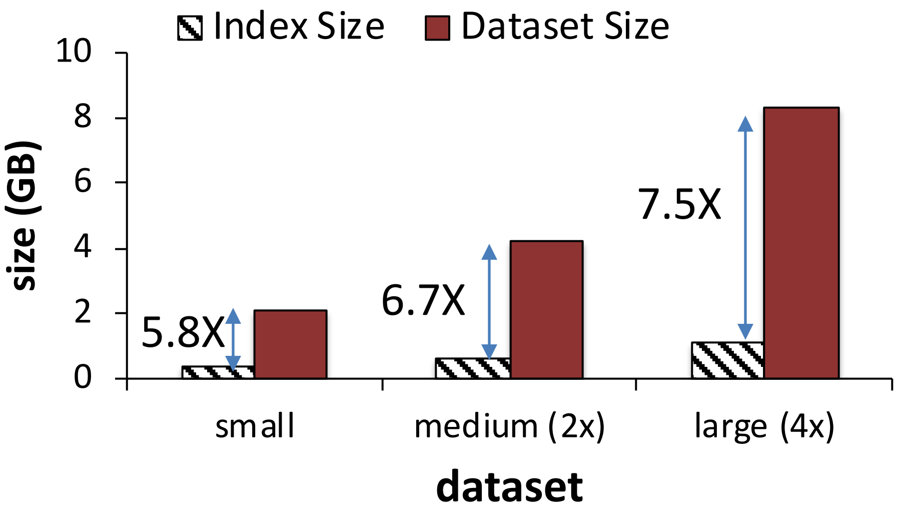
9X to 23X speedup

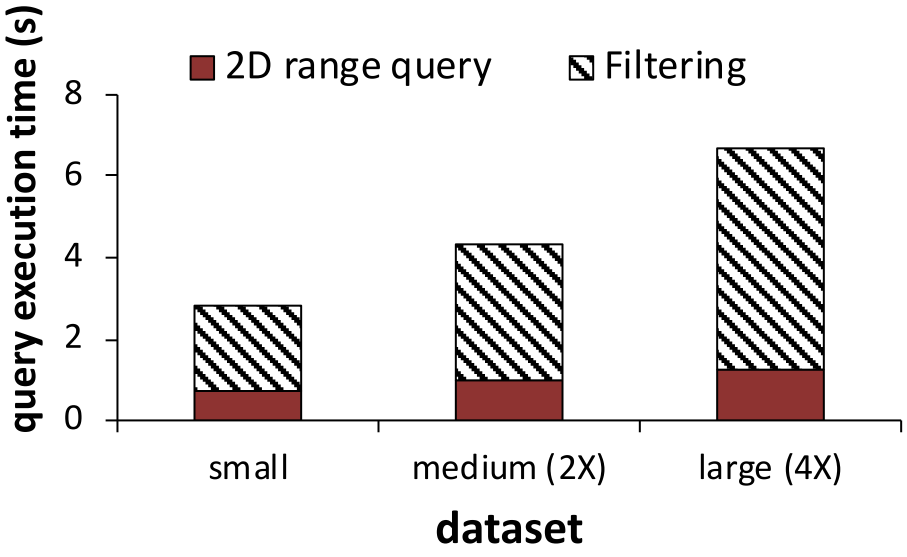
RUBIK Sensitivity Analysis

Configuration: 128 bins

Datasets: 500K – 2M time series, 1024 time steps, 2.1GB – 8.4GB

Benchmark: 60 threshold queries, random thresholds, up to 15% selectivity





Increased similarity ⇒ Increased compression

~80% of the time is spent on filtering

Threshold Queries on Time Series

- Subsets of interest in neuroscience simulations

- **RUBIK** outperforms state-of-the-art by using:
  - Quadtree decomposition

    ⇒ Transformation into a 2D bitmap problem
  - Time series clustering

    ⇒ Similarities across time series are exploited

- **RUBIK** scales particularly well with time series from increasingly detailed simulation models

Thank you!