Visualization data structures

CS524: Big Data Visualization & Analytics

Fabio Miranda

https://fmiranda.me

UIC COMPUTER SCIENCE

Data structures for visualization

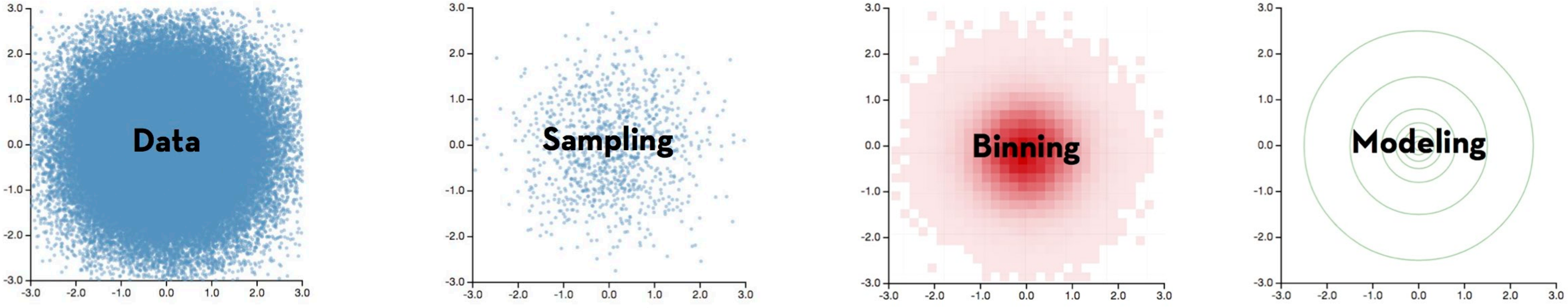
- immense [Liu et al., 2013]
- Nanocube [Lins et al., 2013]
- TopKube [Miranda et al., 2018]
- Time Lattice [Miranda et al., 2019]
- Learned cubes
- …

Visualization requirements

- Visualizations have a specific set of “interaction patterns” and resolutions.
- Data schemes need to be aware of these limitations.

Perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not number of records.

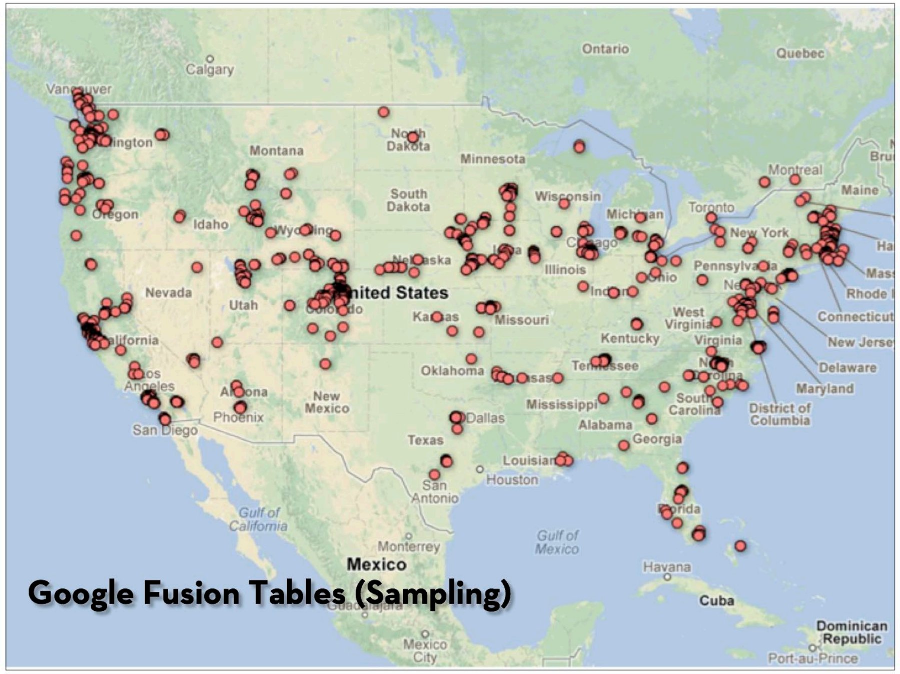
Data reduction techniques

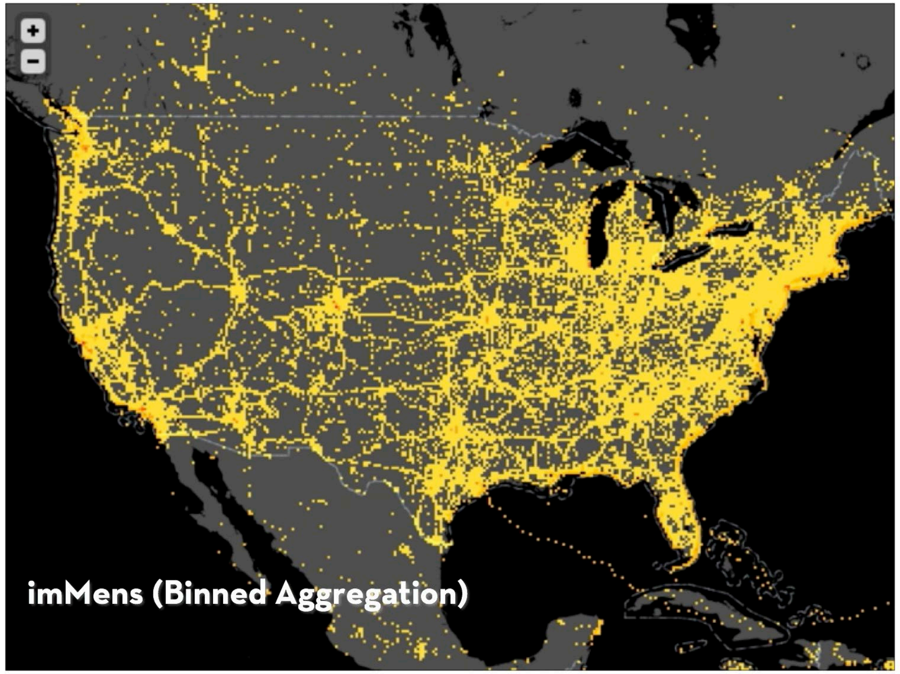


Different reduction techniques applied on the same dataset

[Liu et al., 2013]

Data reduction techniques





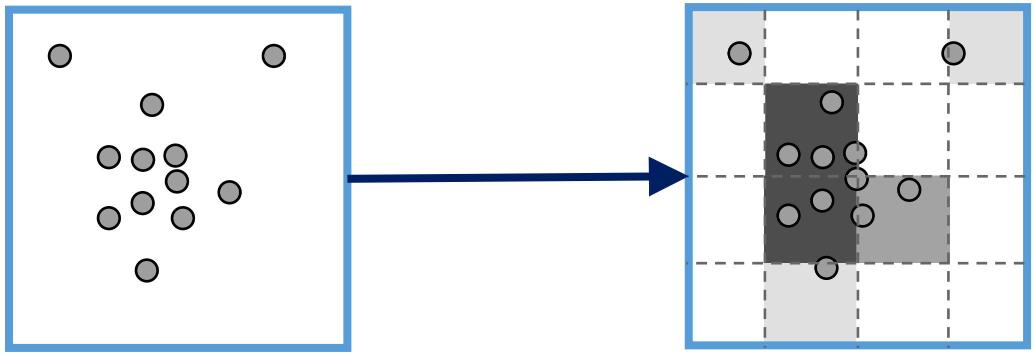
[Liu et al., 2013]

Data binning

Bin	Aggregate	Smooth	Plot
Divide data domain into discrete buckets / cells	Aggregate data (count, sum, average, min, max)	Smooth aggregates (optional)	Visualize the aggregate summaries

Binning

- Divide data domain into discrete buckets / cells
  - Categorical data: already discrete
  - Numbers: choose bin intervals
  - Temporal: choose time unit (hour, day, month, …)
  - Spatial: bin x, y coordinates after cartographic projection



Aggregation

- Aggregate the data to crate a summary, grouped by chosen bins.
- Different classes of aggregation functions:
  - <u>Distributive</u>: functions defined by structural recursion.
  - <u>Algebraic</u>: functions expressed by finite algebraic expressions defined over distributive functions (decomposable aggregate functions).
  - <u>Holistic</u>: all other functions

Aggregation

- Distributive:

$$COUNT(x) = 1$$
$$COUNT(X \uplus Y) = COUNT(X) + COUNT(Y)$$

$$SUM(x) = x$$
$$SUM(X \uplus Y) = SUM(X) + SUM(Y)$$

$$MAX(x) = x$$
$$MAX(X \uplus Y) = max(MAX(X), MAX(Y))$$

Aggregation

- Algebraic:

$$AVG(X \uplus Y) = \frac{SUM(X) + SUM(Y)}{COUNT(X) + COUNT(Y)}$$

$$SUM(X^2 \uplus Y^2) = SUM(X^2) + SUM(Y^2)$$

$$STDDEV(x) = \sqrt{\frac{SUM(x^2)}{COUNT(x)} - AVG(x)^2}$$

$$STDDEV(X \uplus Y) = \sqrt{\frac{SUM(X \uplus Y)}{COUNT(X \uplus Y)} - AVG(X \uplus)^2}$$

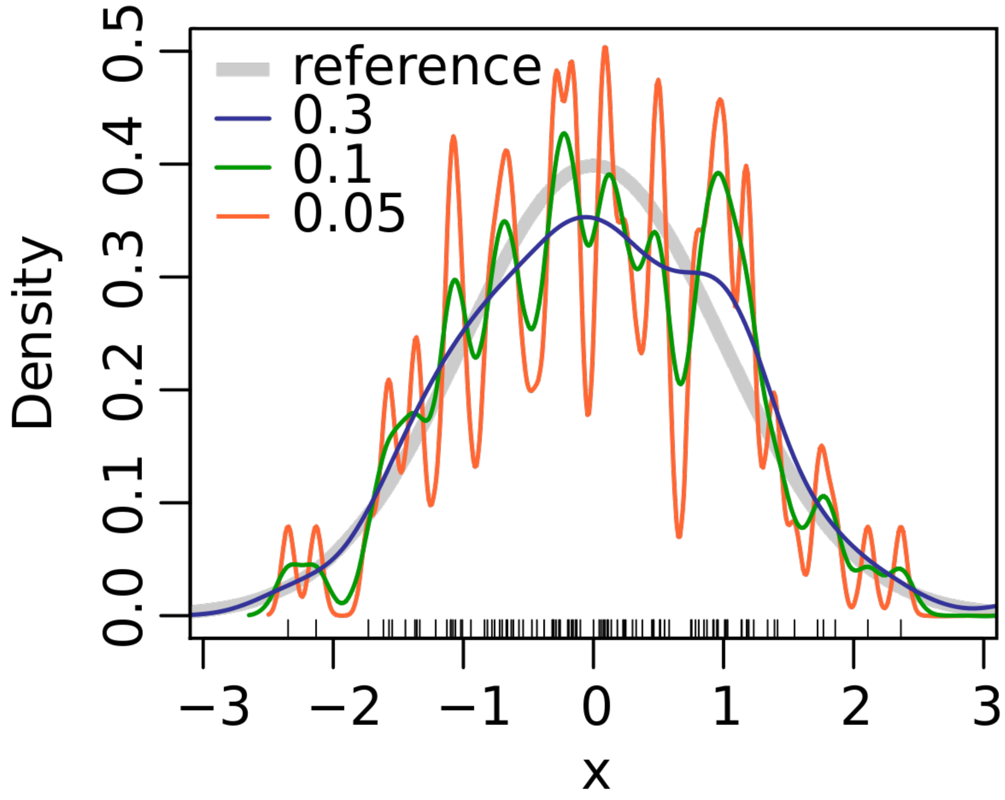
$$\sigma = \sqrt{E[(X-\mu)^2]}$$
$$= \sqrt{E[X^2] - (E[X]^2)}$$

Aggregation

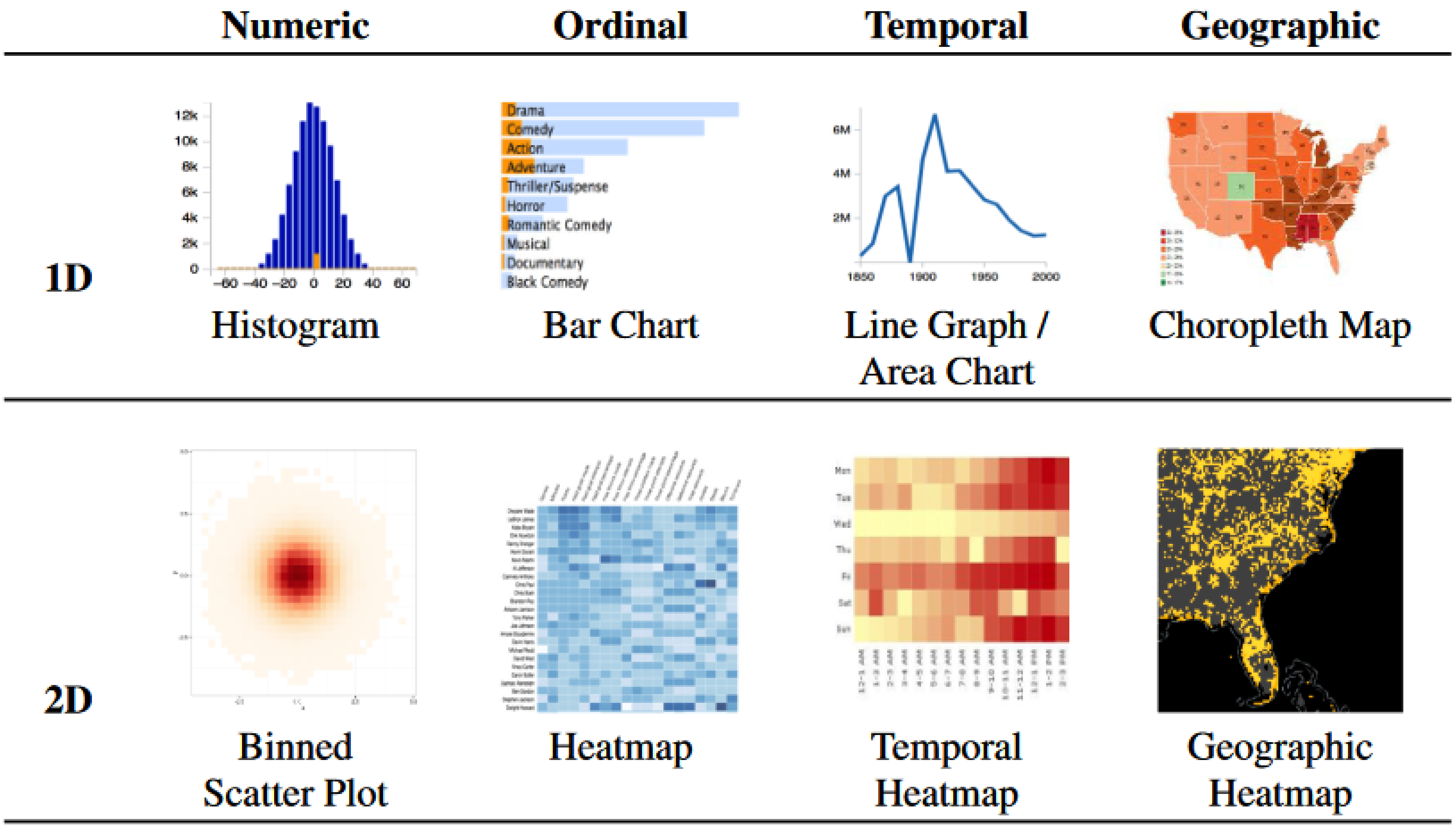
- Holistic: all other functions, no bound on the size of the storage needed to describe sub-aggregate.
  - Most frequent
  - Median
  - …

Smoothing

- Perform smoothing (e.g., KDE) on aggregated data to better approximate underlying continuous density:
  - Large bandwidth: coarse density with little details.
  - Small bandwidth: too much detail and not general enough to cover new or unseen examples.

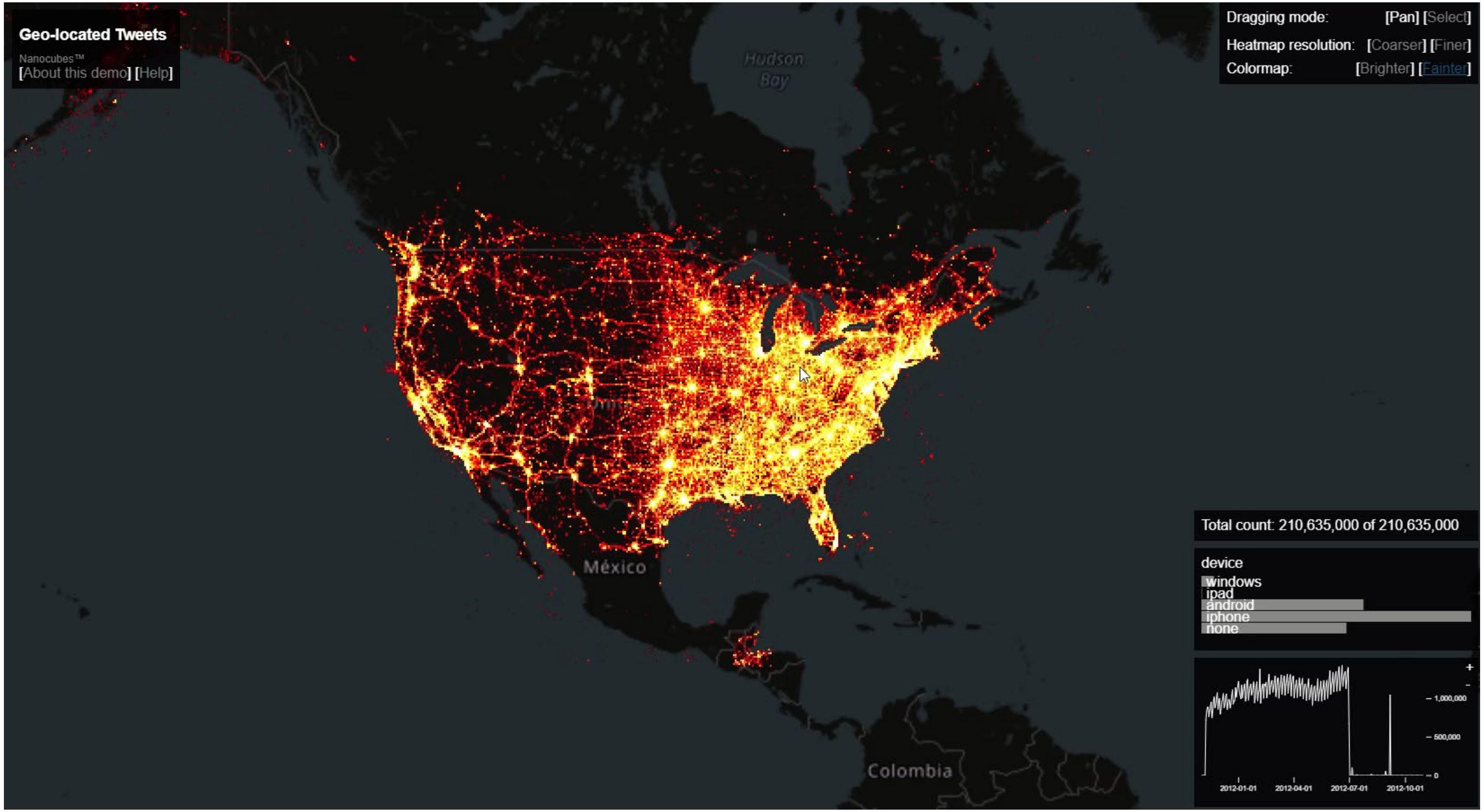


Visualization designs for binned plots

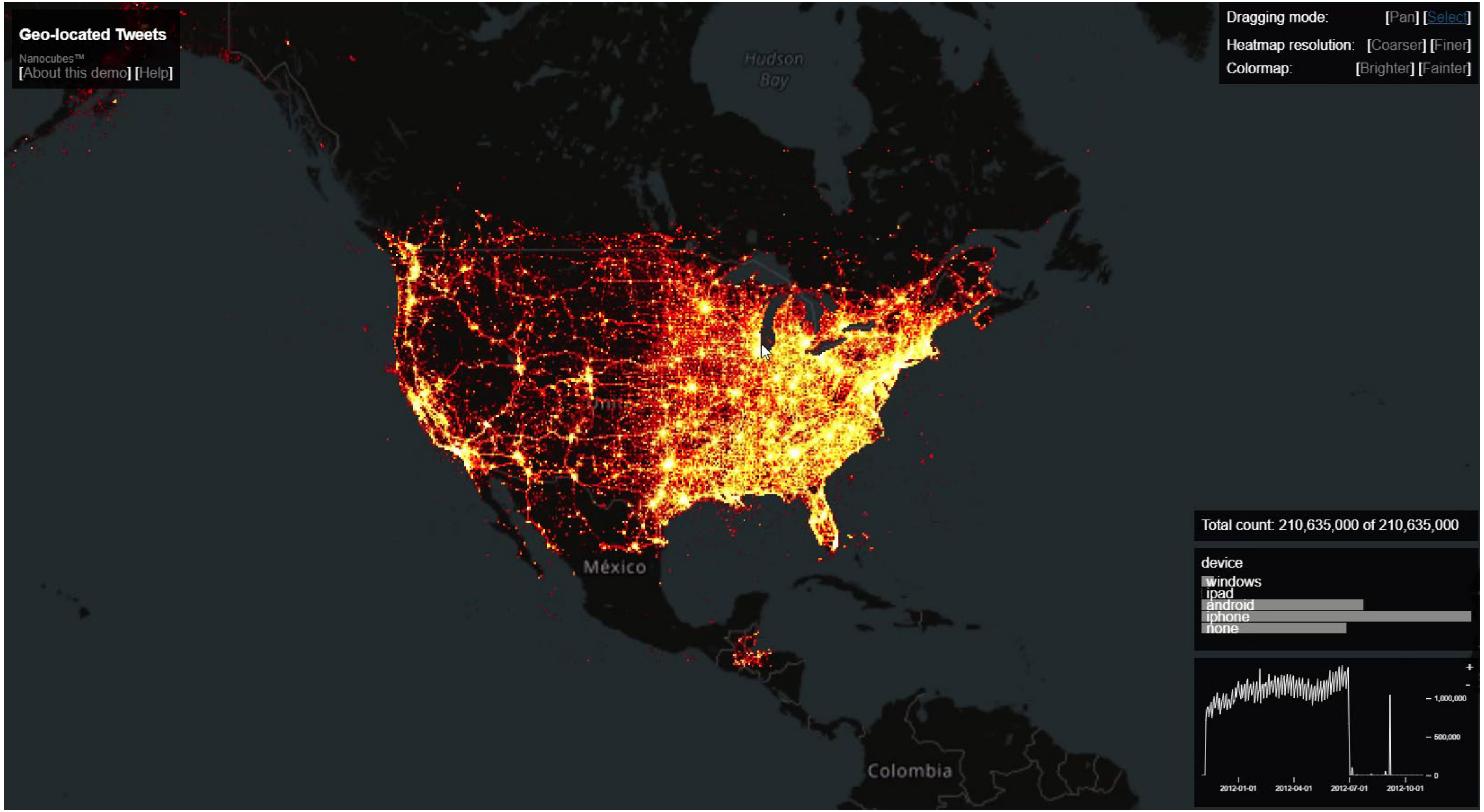


[Liu et al., 2013]

Visualization requirements: brushing & linking

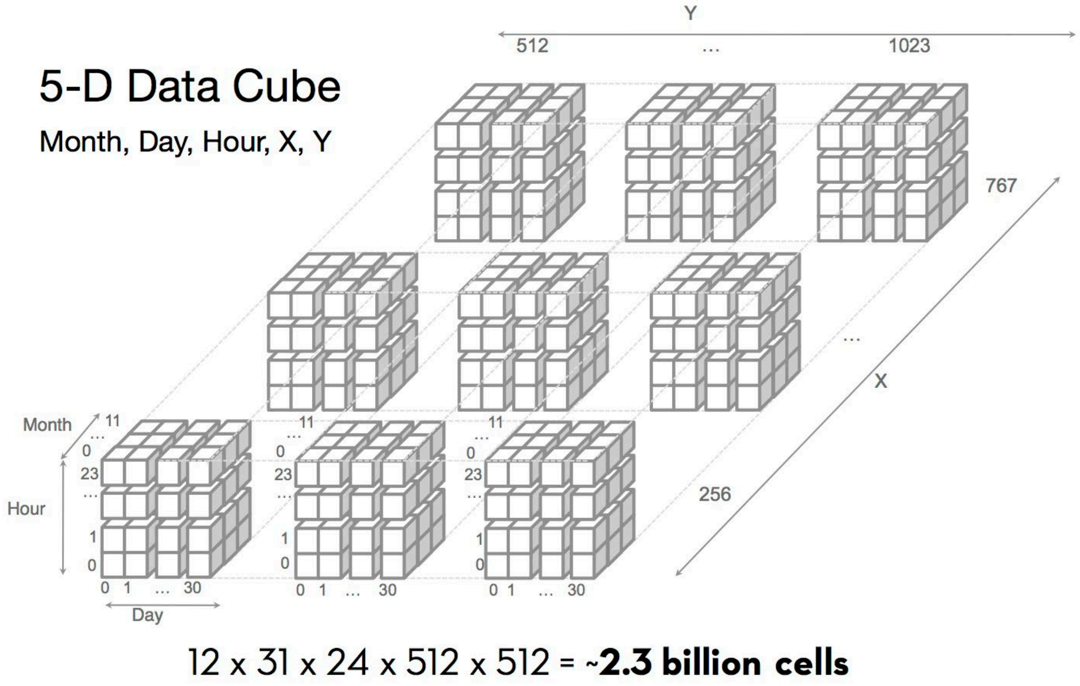


Visualization requirements: spatial brushing & linking



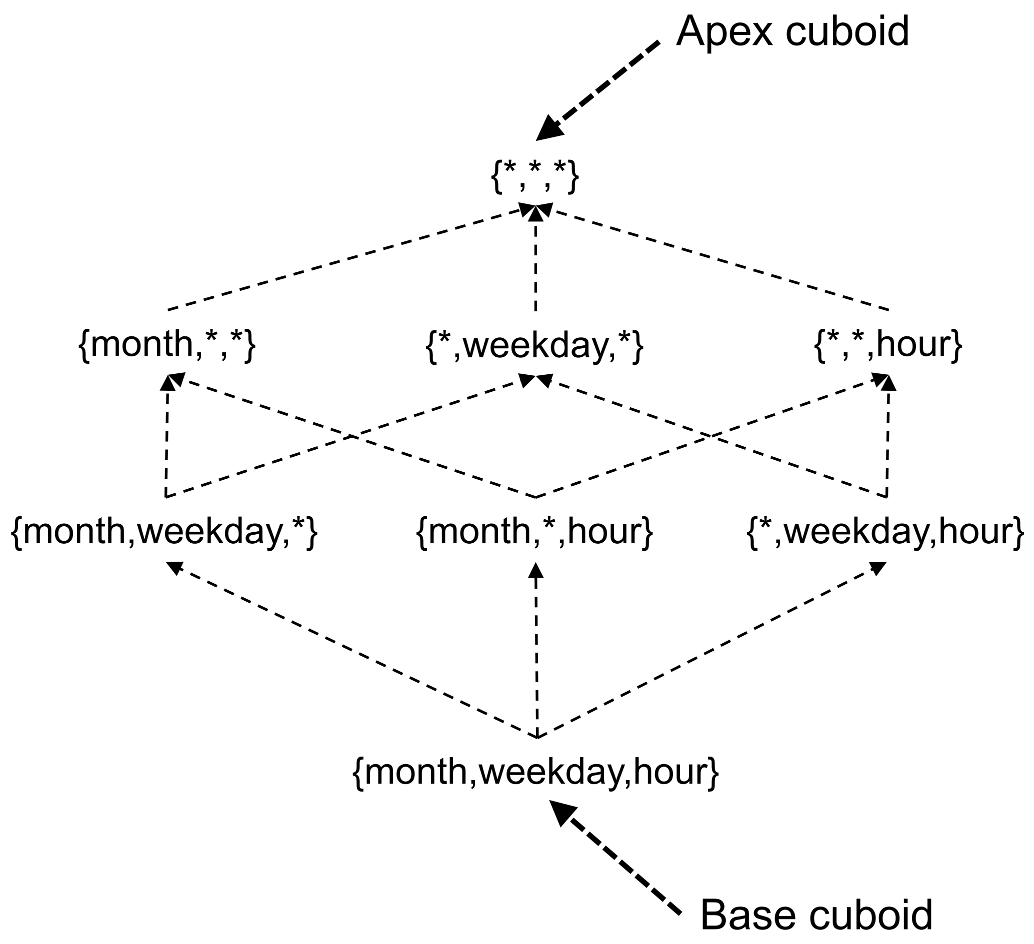
Datacube

- A datacube stores *all* combinations of bins across dimensions.
- Considering the following binning schema:
  - Temporal: 12 (months), 31 (days), 24 (hours)
  - Spatial: 512 x 512
  - Total: ~2.3 billion cells (!!!)



Datacube

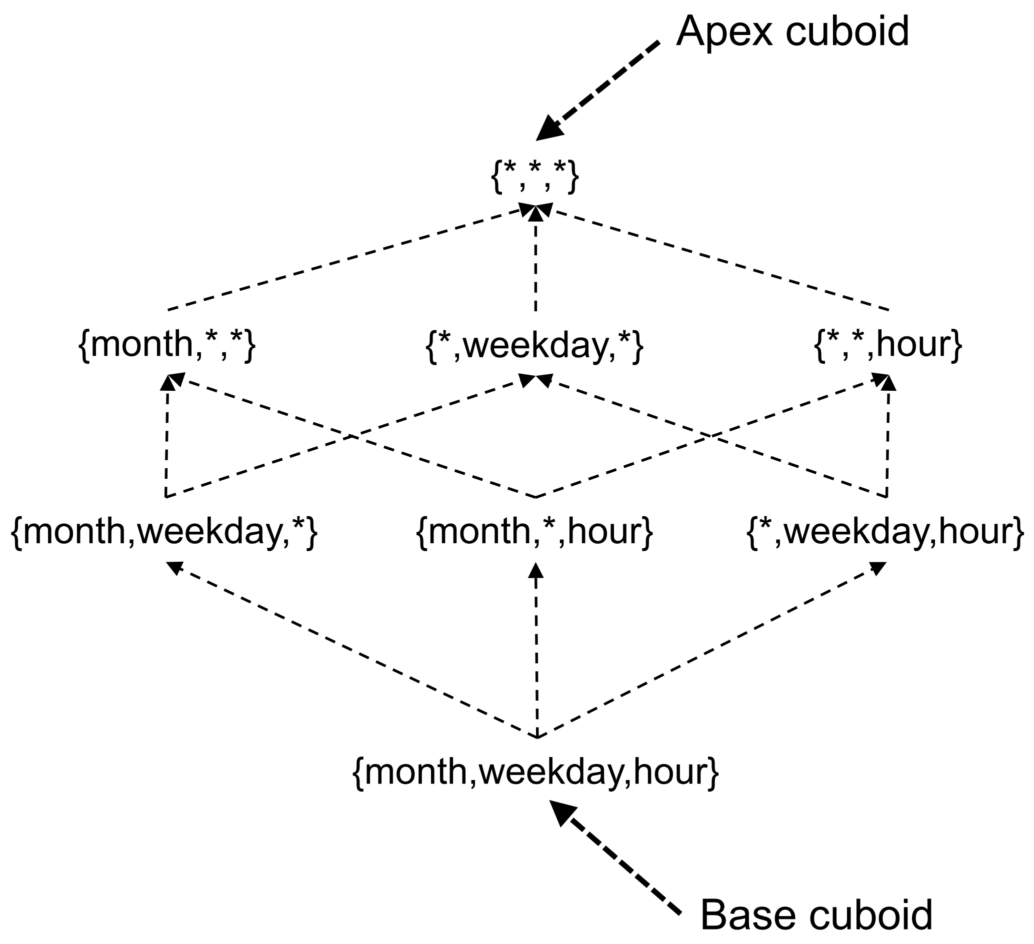
- Datacube can be viewed as a lattice of cuboids:
  - Top cuboid contains only one cell.
  - Base cuboid contains all cells.
- Materialization:
  - Full: materialize every cuboid.
  - Partial: trade-off between storage space and response time.
  - None: on-the-fly aggregation.



Total number of cuboids: 2^n, with n dimensions

Datacube

- Elements of a dimension can be stored as a hierarchy:
  - Set of parent-child relationships where the parent summarizes its children.
  - Block → City → State → Country



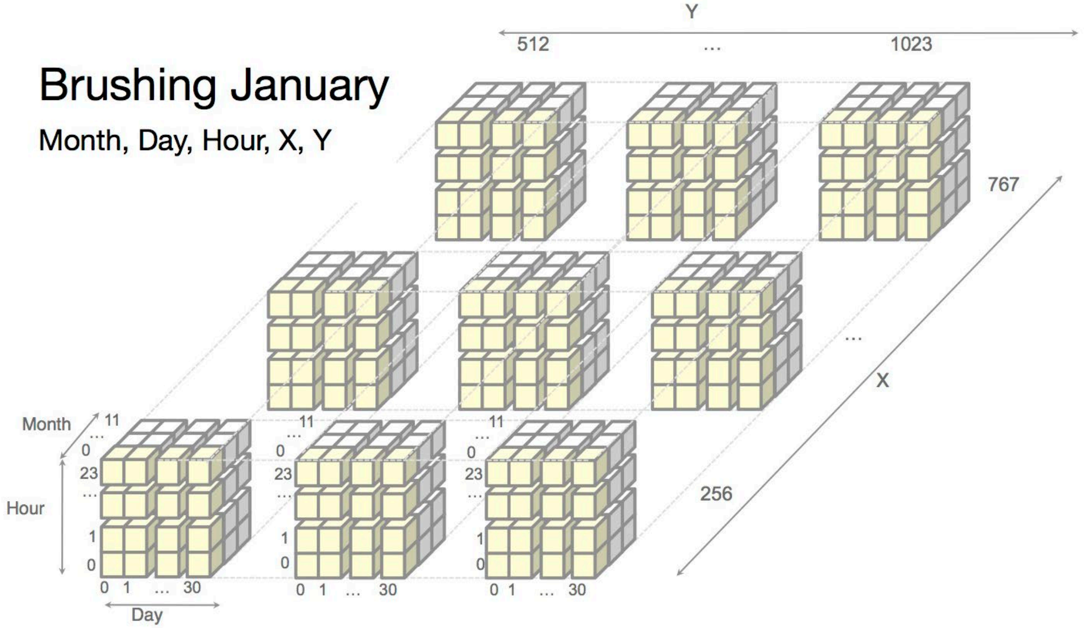
Total number of cuboids: 2^n, with n dimensions

Slice, dice, drill down, roll up

- Slice: selects one particular dimension from a cube, providing a new sub-cube.
- Dice: selects two or more dimensions from a cube, providing a new sub-cube.
- Drill down: descending the hierarchy.
- Roll up: data is aggregated by ascending the hierarchy (e.g., level of a city to the level of a country).

imMens

- Performing operations over cuboids can be prohibitively expensive.
  - E.g., Aggregating over the month of January will result in performing computation over 1/12th of the cube.

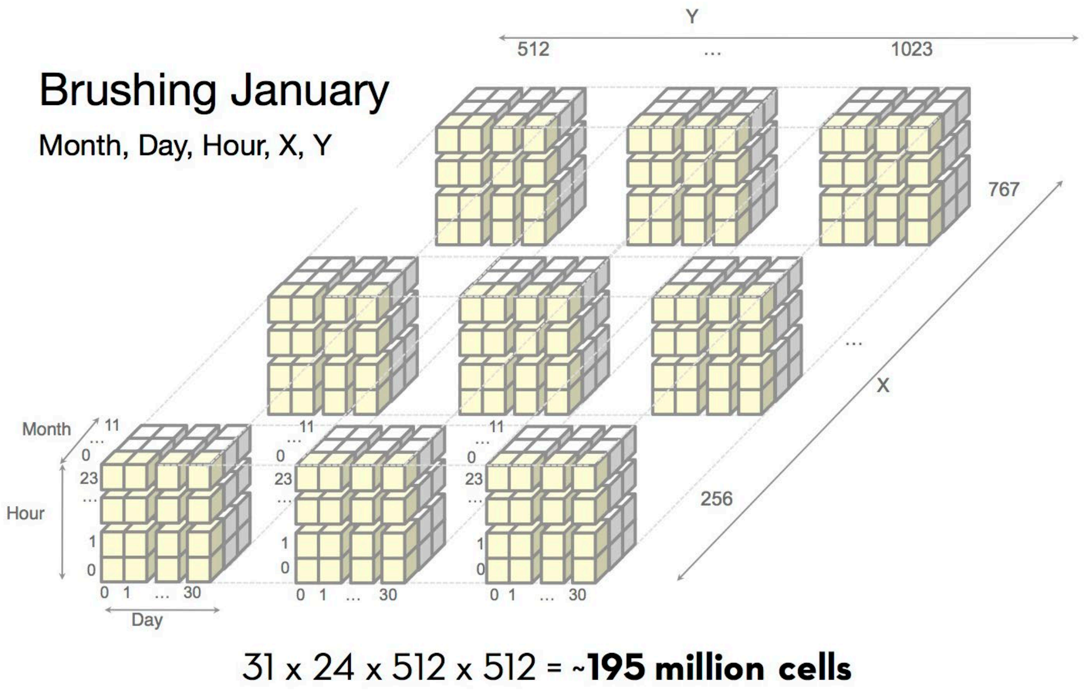


31 x 24 x 512 x 512 = ~**195 million cells**

[Liu et al., 2013]

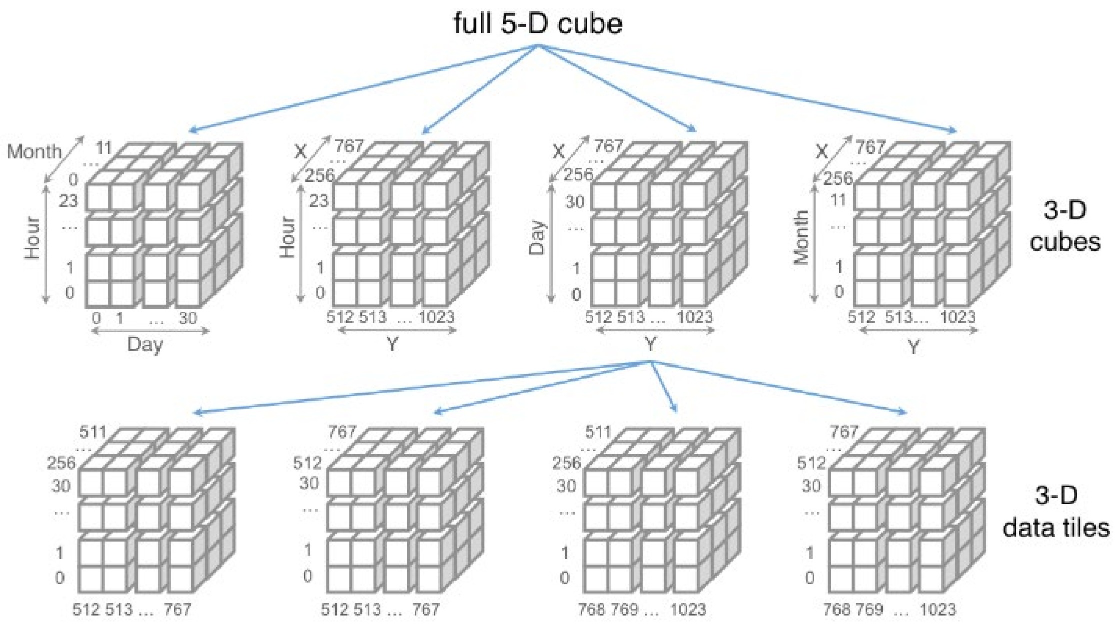
imMens

- imMens addresses this issue by precomputing image tiles with aggregations.
- Like OSM tiles, but now storing data: multivariate data tiles.
- How to solve the combinatorial explosion of multiple dimensions?



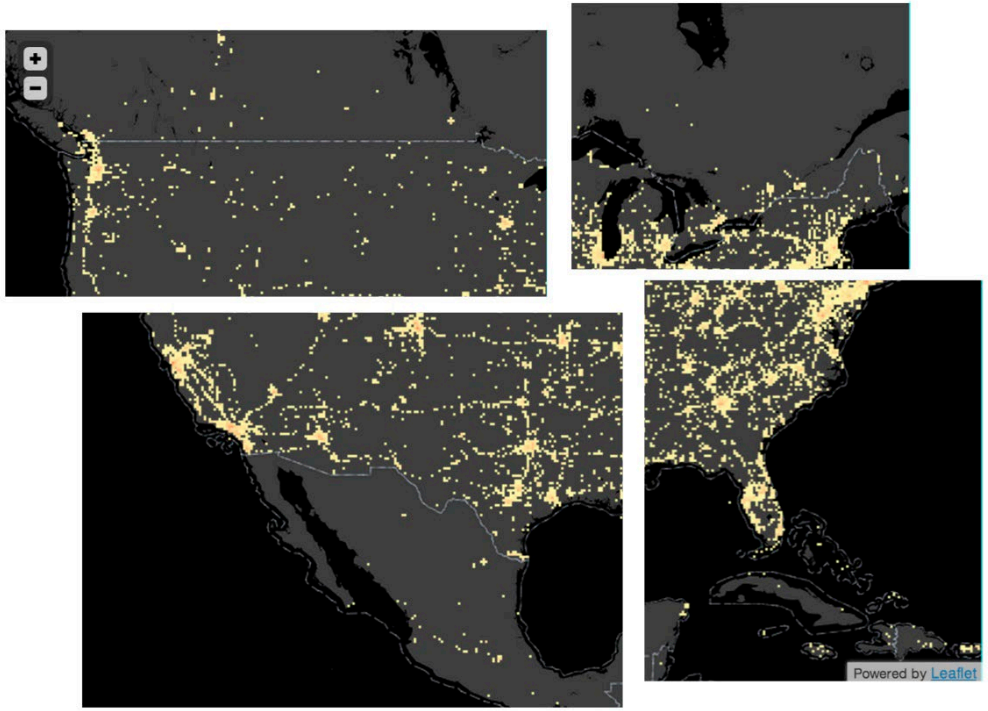
imMens

- For a pair of binned plots, maximum number of dimensions needed to support brushing & linking is four.
  - Between two binned scatterplots that do not share a dimensions.
- Idea: decompose the full cube into a collection of smaller 3- or 4-dimensional projections.



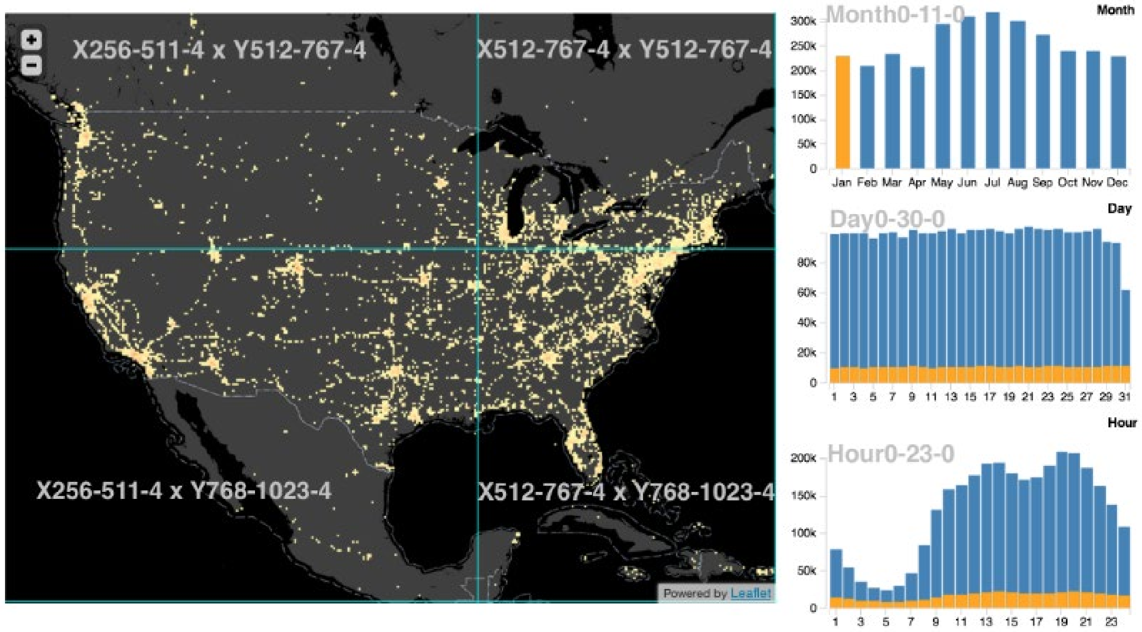
imMens

- Certain dimensions may still require a large number of bins.
  - Spatial dimension: many bins to represent the entire globe.
  - Handle this by breaking up these projections by index ranges (similar to OSM tiles).



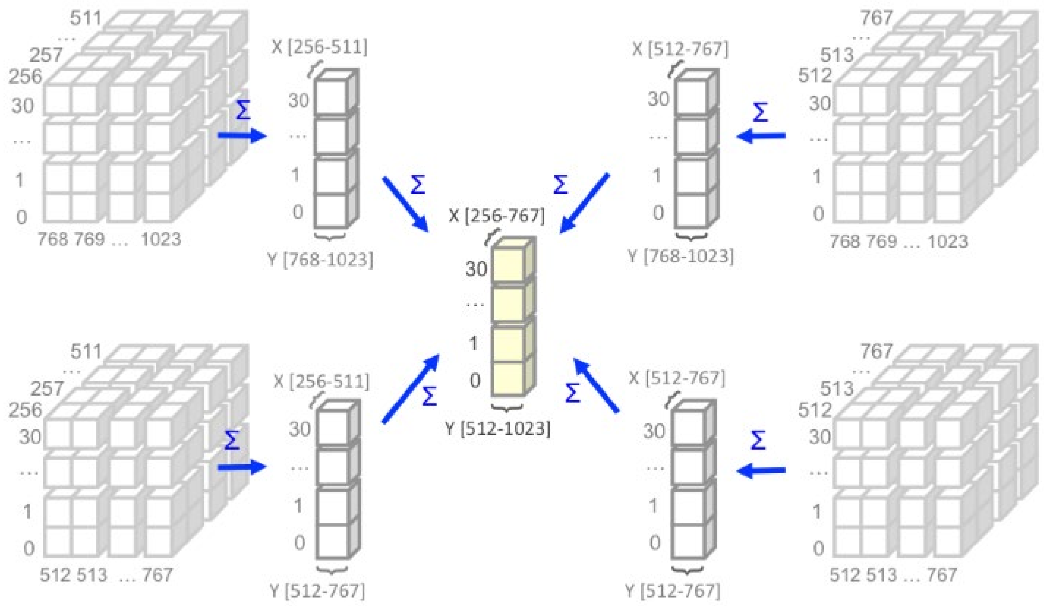
imMens

- Brightkite example:
  - Four dimensions – space, month, day, hour.
  - Space: 4 512x512 tiles.
  - Time: 1 tile each.
  - Supporting brushing & linking:
    - Space – month: 4 x 1 tiles
    - Space – day: 4 x 1 tiles
    - Space – hour: 4 x 1 tiles
    - Month – day – hour: 1 tile
    - Total: 13 tiles

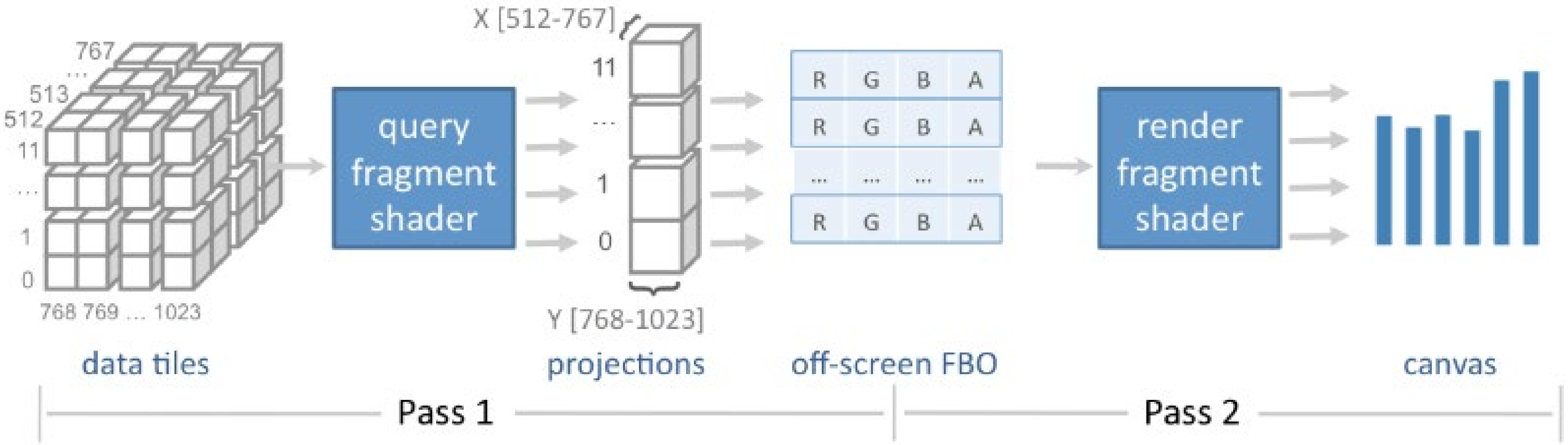


imMens

- Brushing & linking involves aggregating data tiles.
  - User selects a region → compute aggregation → highlight corresponding histograms.



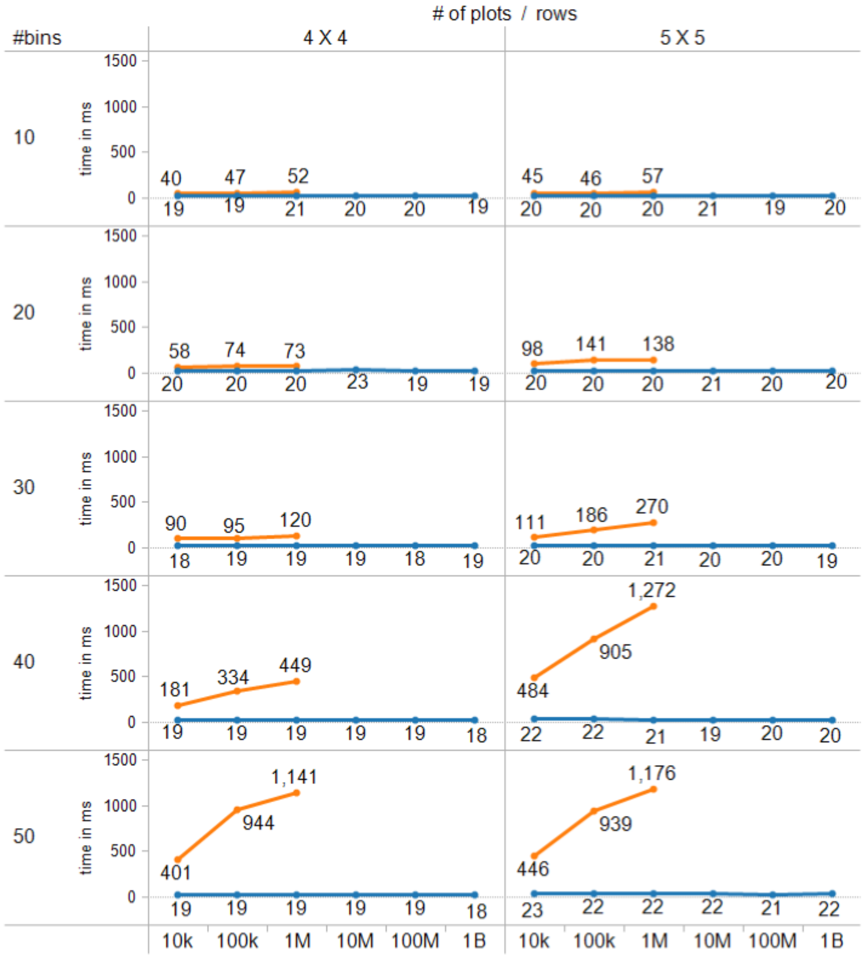
imMens



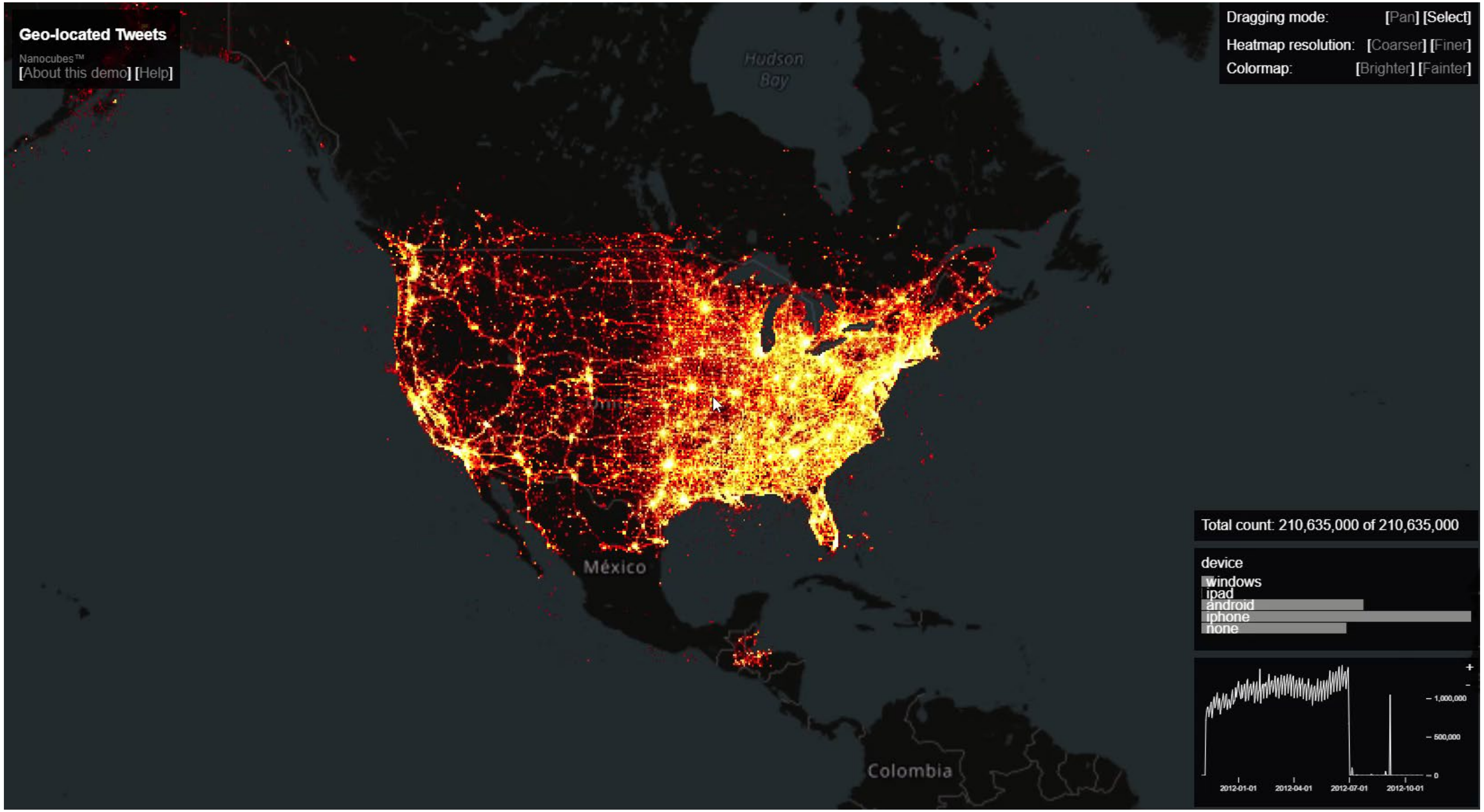
Two-pass approach for parallel data querying and rendering using WebGL fragment shaders.

imMens

Constant query time, even for datasets with 1B data points.

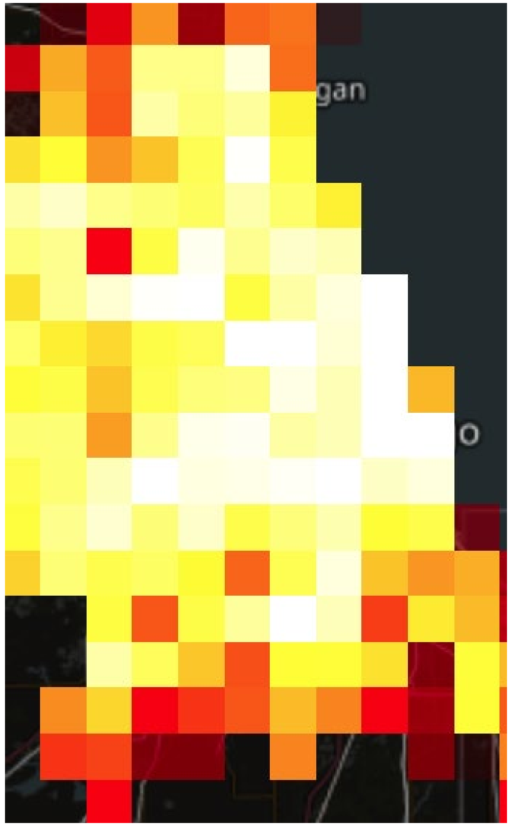


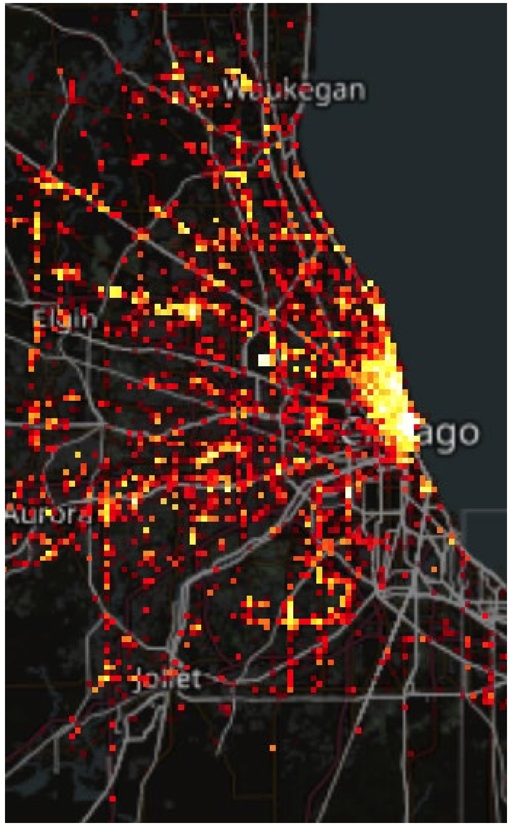
Visualization requirements: panning and zooming



Nanocubes

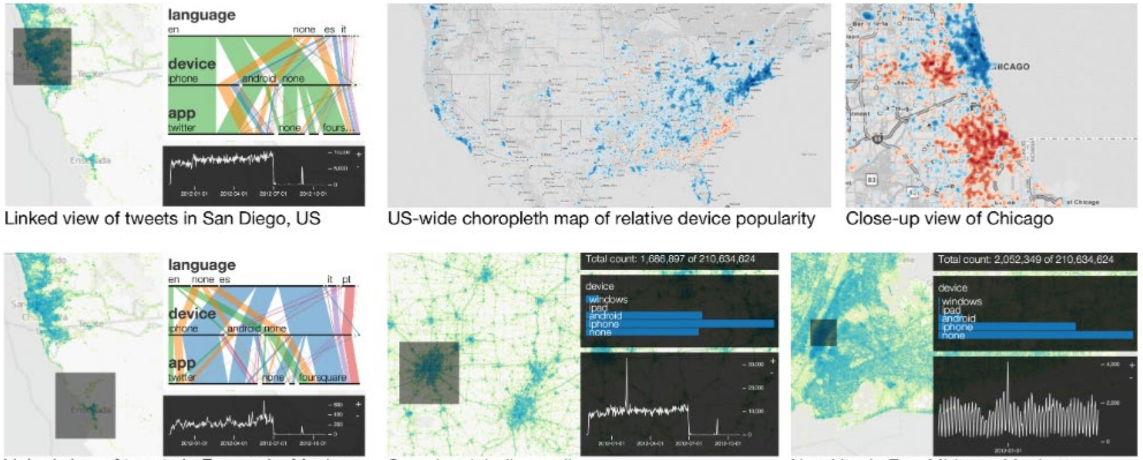
- How to handle different spatial and temporal resolutions?
  - imMens supports “overview first, zoom and filter, details-on-demand”, but <u>not</u> inside spatiotemporal dimensions.
- Trade-off:
  - Fine resolution (small bins / cells): high memory consumption, but highly detailed.
  - Coarser resolution (large bins / cells): low memory consumption, but no details.





Nanocubes

- Data structure that supports drilling down both spatial and temporal dimensions, while maintaining manageable memory usage.
- Sparse data structure that bins and counts data points.
- Important trick: allow for shared links across dimensions, and in the same dimension.



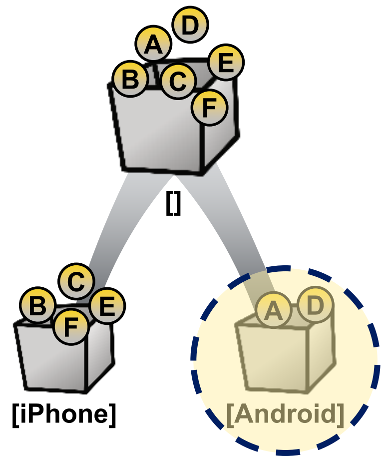
Linked view of tweets in San Diego, US
US-wide choropleth map of relative device popularity
Close-up view of Chicago
Linked view of tweets in Ensenada, Mexico
Superbowl, Indianapolis
New Year's Eve, Midtown Manhattan

COMPUTER SCIENCE

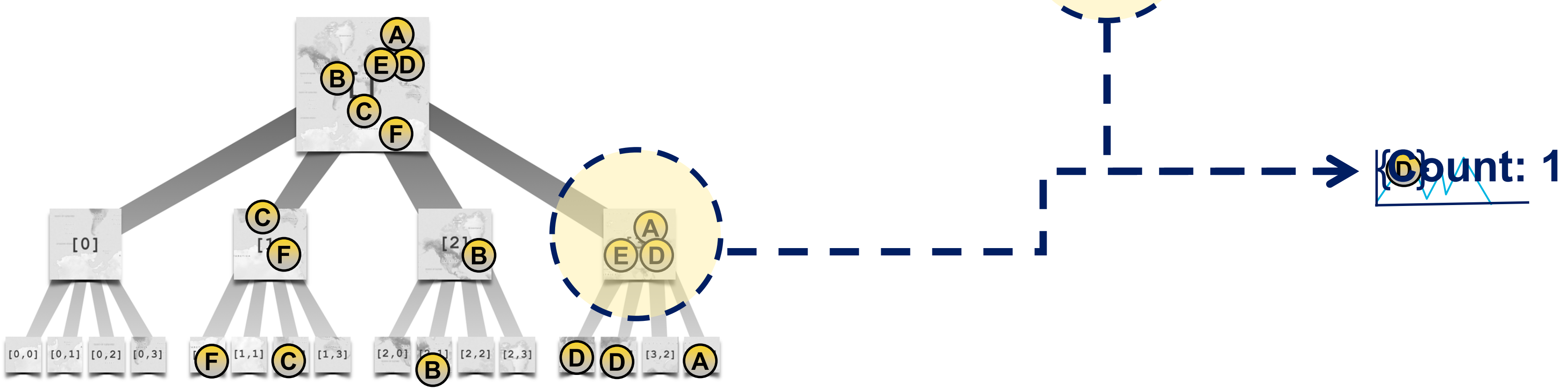
Nanocubes hierarchy

latitude	longitude	keyword	device
42.102908	-73.242852	#phoenix	Android
29.617161	-81.636398	#phoenix	iPhone
23.014051	75.120052	#la	iPhone
26.014051	75.120052	#nyc	Android
28.014051	74.120052	#la	iPhone
23.014051	75.120052	#phoenix	iPhone

	device
A	Android
B	iPhone
C	iPhone
D	Android
E	iPhone
F	iPhone

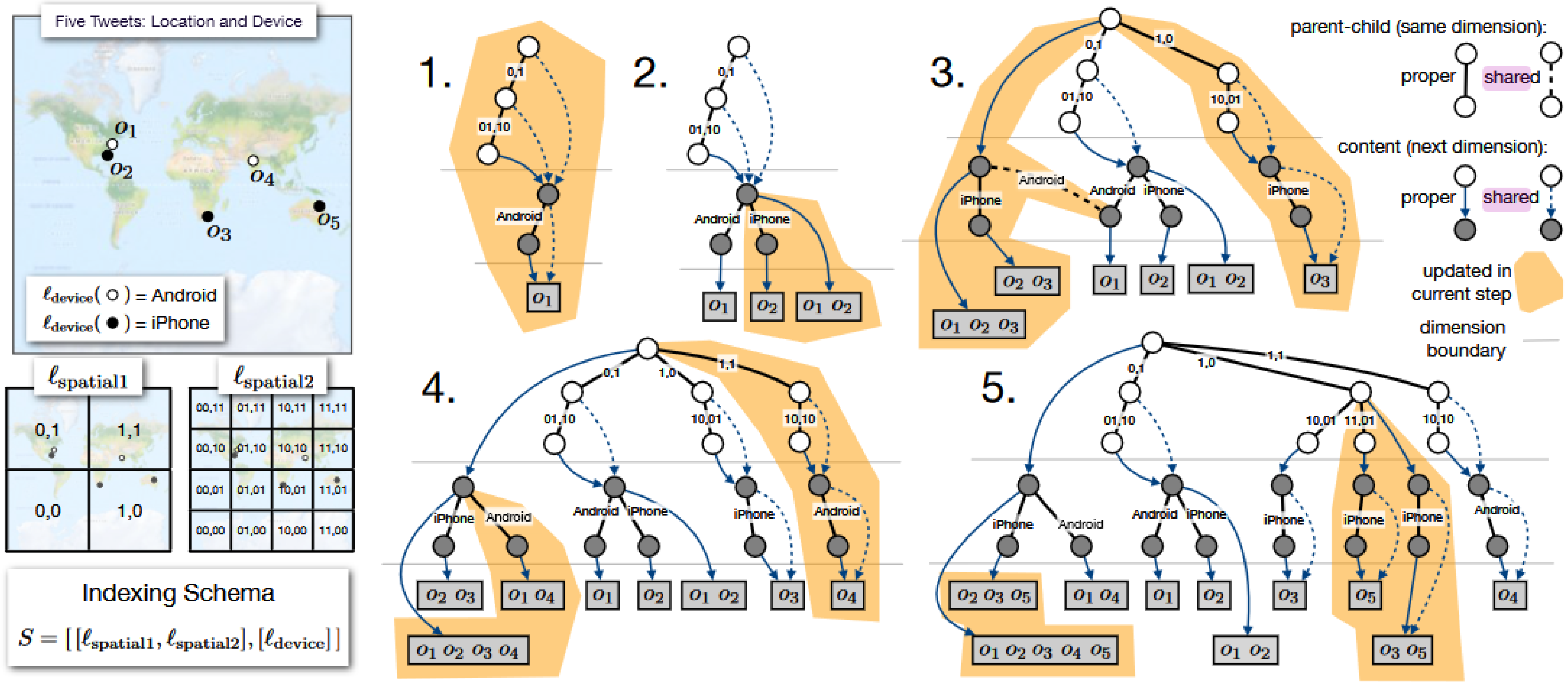


	latitude	longitude
A	42.102908	-73.242852
B	29.617161	-81.636398
C	23.014051	75.120052
D	26.014051	75.120052
E	28.014051	74.120052
F	29.61161	-81.63638



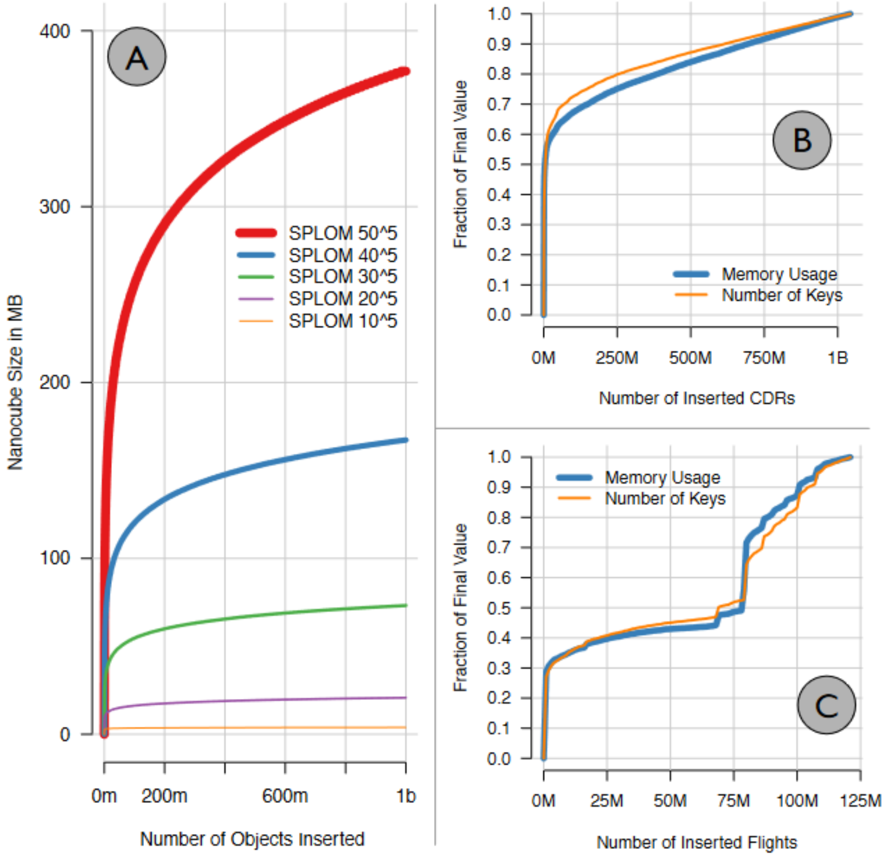
Nanocubes: shared links

- Important trick: allow for shared links across dimensions, and in the same dimension.
- Sharing is responsible for significant memory savings.



Experiments

- Mean query time was $800\mu s$ (less than 1 millisecond!), with a maximum of 12 milliseconds.
- Memory requirements for Twitter dataset (210 million tweets):
  - Without sharing: 37 GB
  - With sharing: 10 GB



UIC COMPUTER SCIENCE

Spatiotemporal + keywords data

When?

What?

Where?

Spatiotemporal + keywords data

Density map Timeseries

Opportunity to not only explore ***where*** and ***when*** things happen, but also ***what*** is happening

Top Keywords

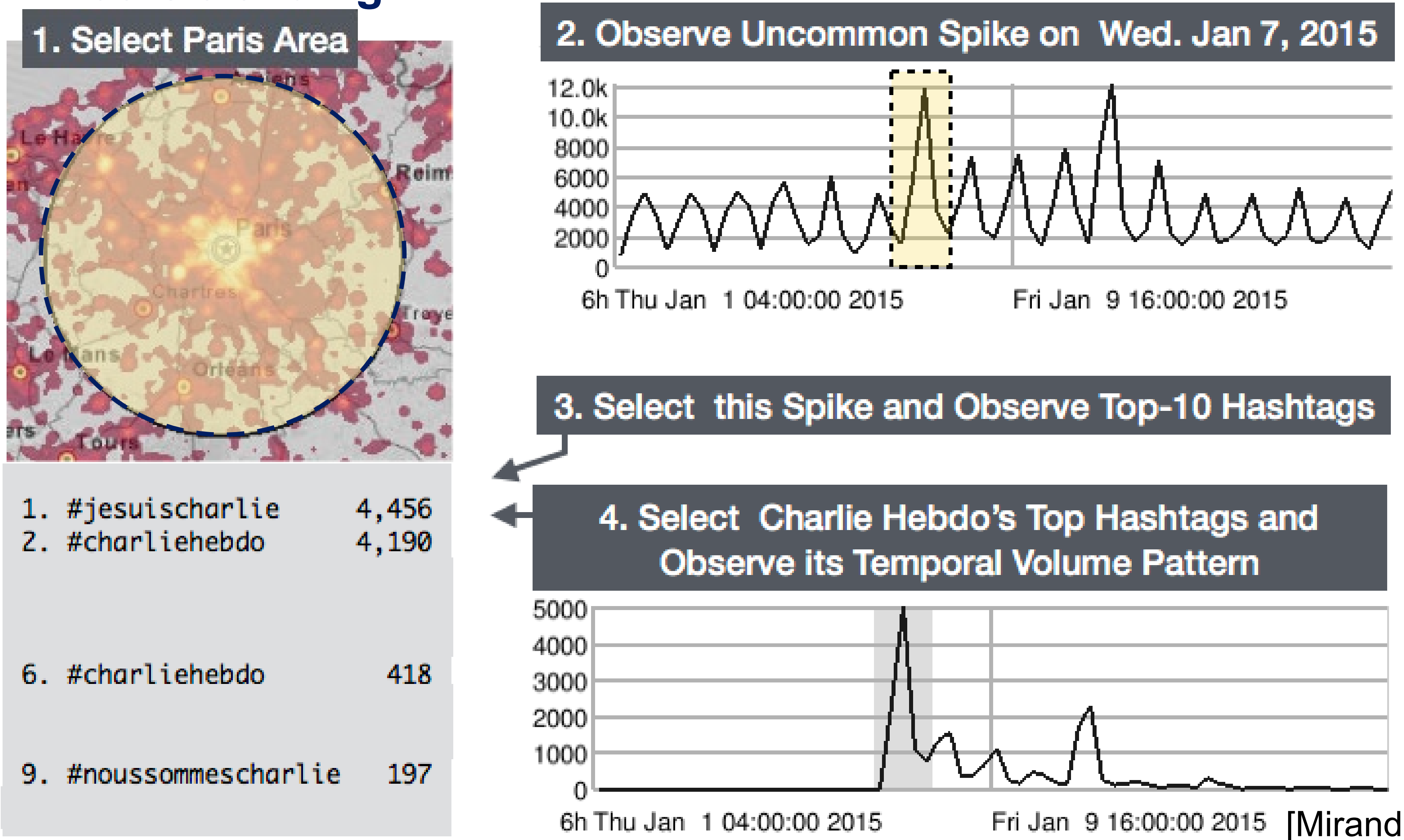
Top-k queries

- Top-k: **k** most popular keywords.
  - Compute list of most popular hashtags in a given space and time from a **large** set of data points.
- How a set of keywords changes over space and time?
  - Where and when certain hashtags are popular.
- How the volume of keywords compare with each other?
  - #patriots vs #giants.

Top-k: what is trending?

latitude	longitude	time	keyword
42.102908	-73.242852	Sept 29 2016 00:00	#patriots
29.617161	-81.636398	Sept 29 2016 00:05	#giants
-21.527420	32.493101	Sept 29 2016 00:10	#jets
26.014051	75.120052	Sept 29 2016 00:22	#patriots
-22.698453	145.080990	Sept 29 2016 00:31	#giants

Top-k: what is trending?

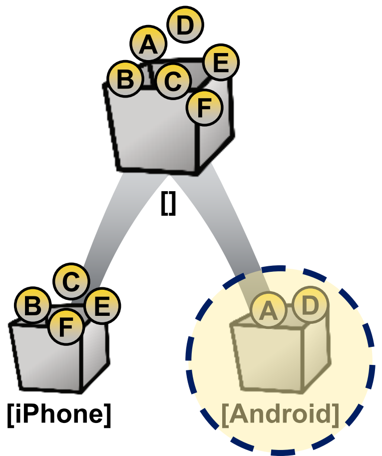


[Miranda et al., 2018]

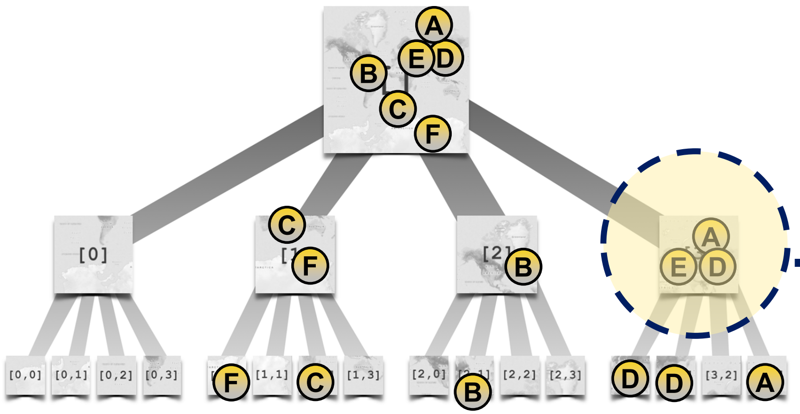
TopKube: A rank-aware datacube

latitude	longitude	keyword	device
42.102908	-73.242852	#phoenix	Android
29.617161	-81.636398	#phoenix	iPhone
23.014051	75.120052	#la	iPhone
26.014051	75.120052	#nyc	Android
28.014051	74.120052	#la	iPhone
23.014051	75.120052	#phoenix	iPhone

	device
A	Android
B	iPhone
C	iPhone
D	Android
E	iPhone
F	iPhone



	latitude	longitude
A	42.102908	-73.242852
B	29.617161	-81.636398
C	23.014051	75.120052
D	26.014051	75.120052
E	28.014051	74.120052
F	29.61161	-81.63638



Count: 1

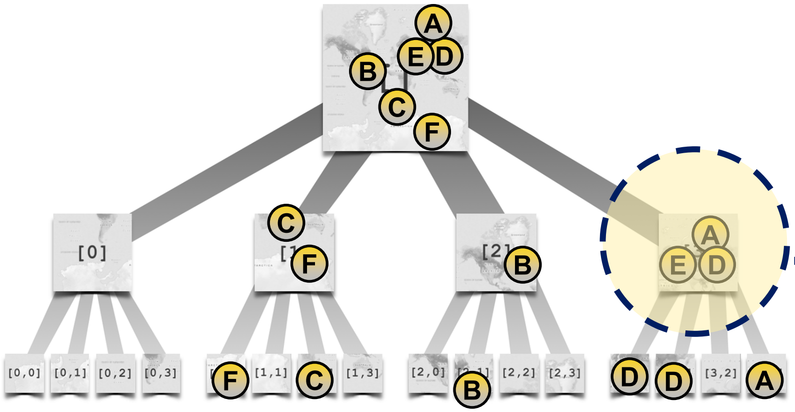
K	c	p
0	10	1
1	22	2
2	15	0

TopKube: A rank-aware datacube

- Following data cube model, aggregate every record along a hierarchy of bins.
- The data structure is a mapping of bins to a pre-computed summary (e.g. count, timeseries).
- In this case, use a ranking summary.

latitude	longitude	keyword
42.102908	-73.242852	#phoenix
29.617161	-81.636398	#phoenix
23.014051	75.120052	#la
26.014051	75.120052	#nyc
28.014051	74.120052	#la
23.014051	75.120052	#phoenix

	latitude	longitude
A	42.102908	-73.242852
B	29.617161	-81.636398
C	23.014051	75.120052
D	26.014051	75.120052
E	28.014051	74.120052
F	29.61161	-81.63638



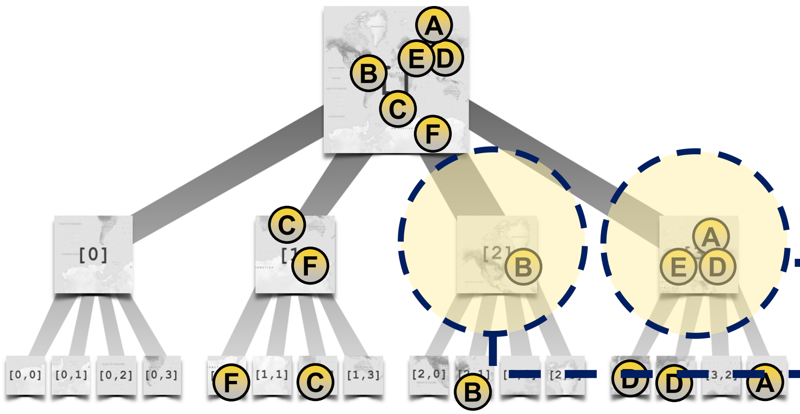
K	c	p
0	10	1
1	22	2
2	15	0

TopKube: A rank-aware datacube

- Following data cube model, aggregate every record along a hierarchy of bins.
- The data structure is a mapping of bins to a pre-computed summary (e.g. count, timeseries).
- In this case, use a ranking summary.

latitude	longitude	keyword
42.102908	-73.242852	#phoenix
29.617161	-81.636398	#phoenix
23.014051	75.120052	#la
26.014051	75.120052	#nyc
28.014051	74.120052	#la
23.014051	75.120052	#phoenix

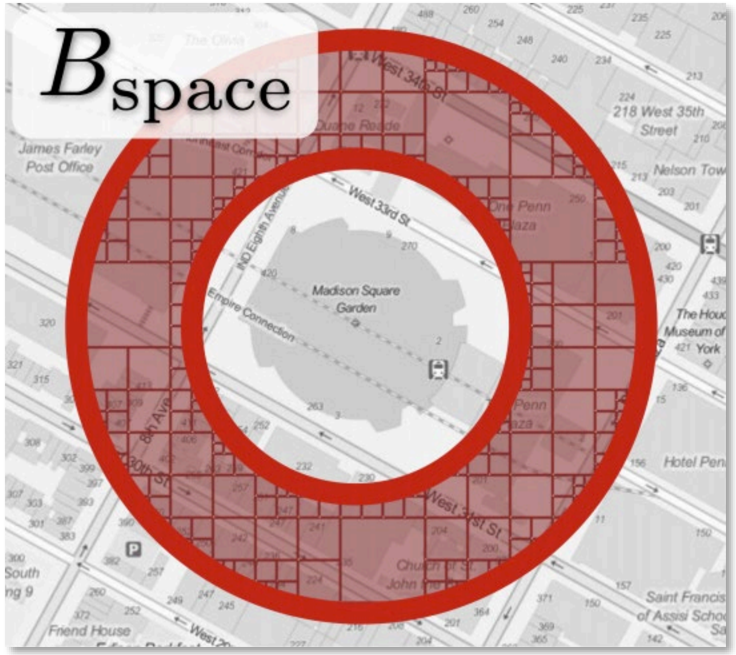
	latitude	longitude
A	42.102908	-73.242852
B	29.617161	-81.636398
C	23.014051	75.120052
D	26.014051	75.120052
E	28.014051	74.120052
F	29.61161	-81.63638



K	c	p
0	10	1
1	22	2
2	15	0

K	c	p
0	10	1
1	22	2
2	15	0

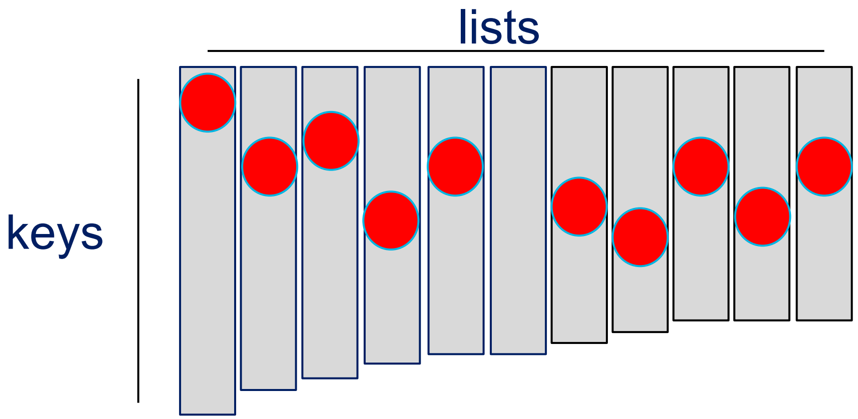
TopKube: A rank-aware datacube



- Spatial and temporal selections usually result in several (thousands) ranking summaries.
- How to efficiently merge them?

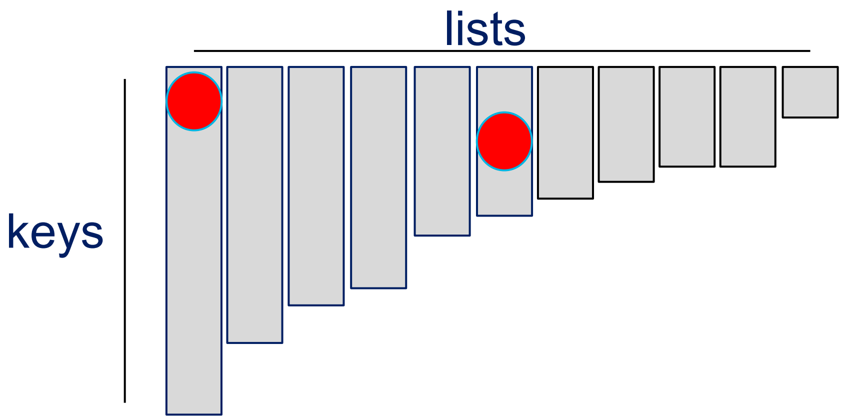
Top-k from ranked lists

Threshold Algorithm [Fagin et al., 2003]



Ideal scenario:

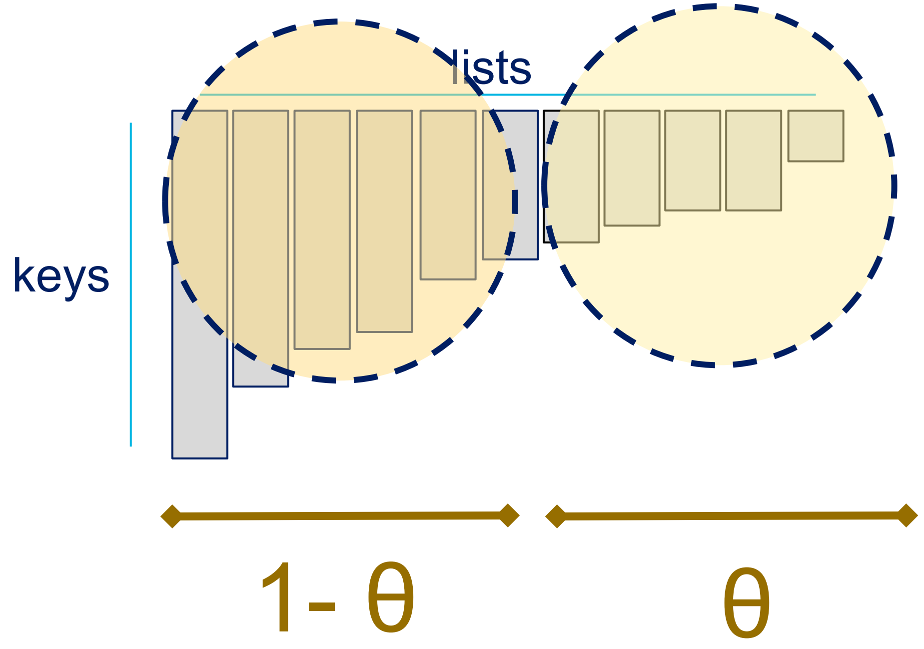
- A key is in almost all lists
- Low number of misses



Most common scenario:

- A key is in very few lists
- Large number of misses

Top-k from ranked lists



TopKube:

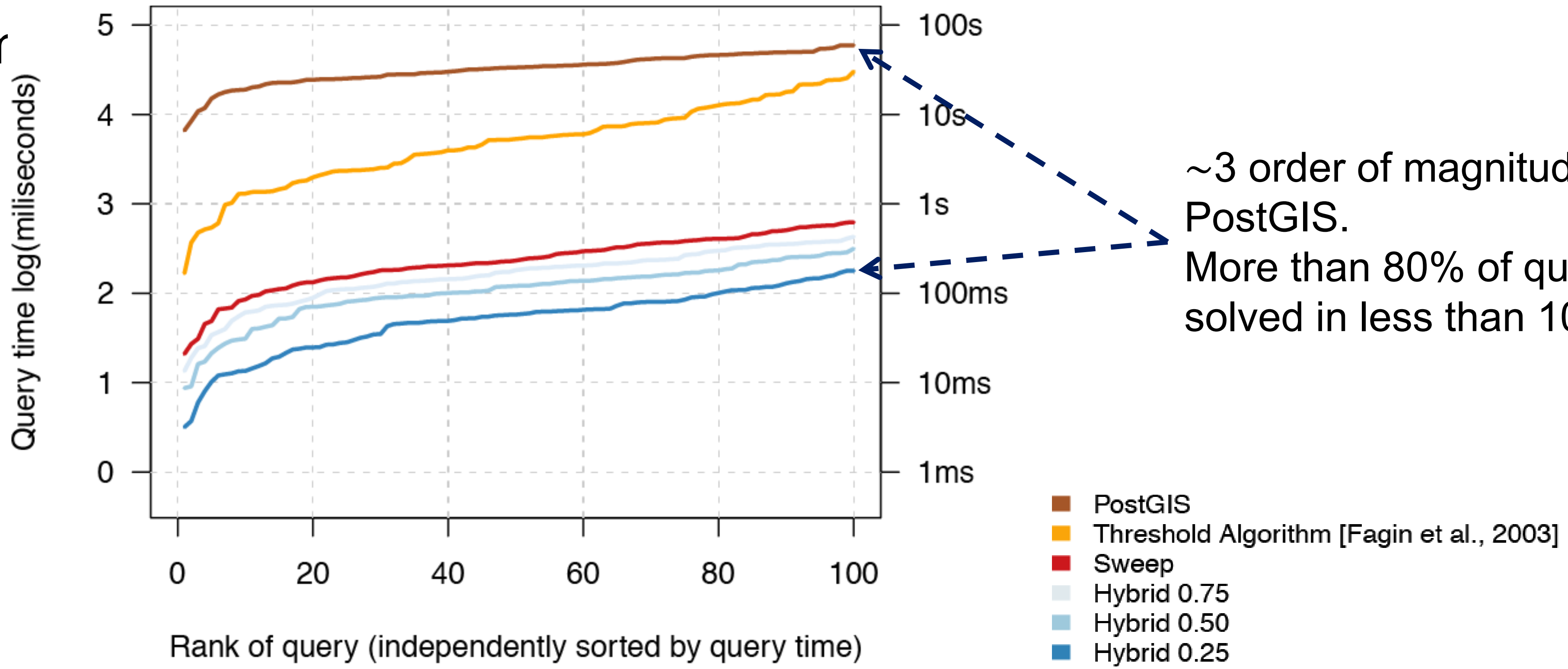
- Run Sweep on α smallest (easiest) problems.
- Run Fagin's Threshold Algorithm on the denser (harder) problems.

Goal:

- Increase key density.
- Decrease the number of wasted searches.

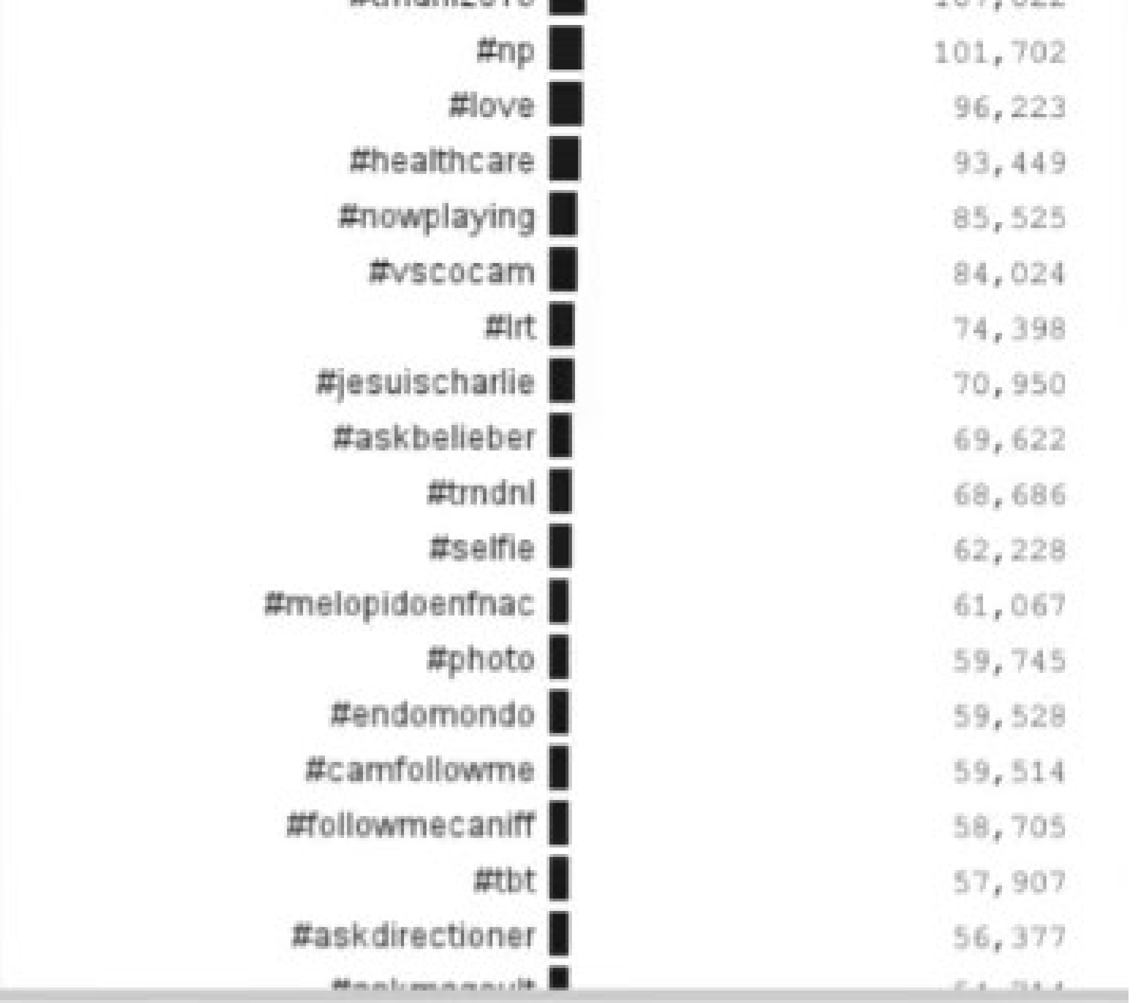
TopKube

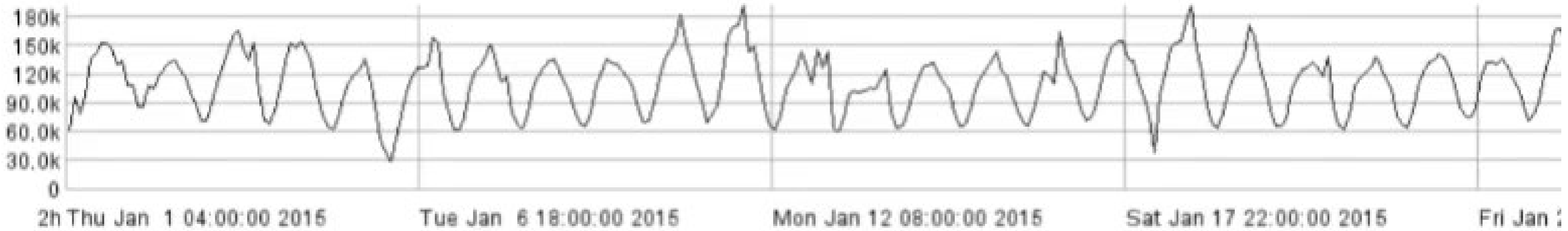
Tim



~3 order of magnitude over PostGIS.
More than 80% of queries solved in less than 100 ms.

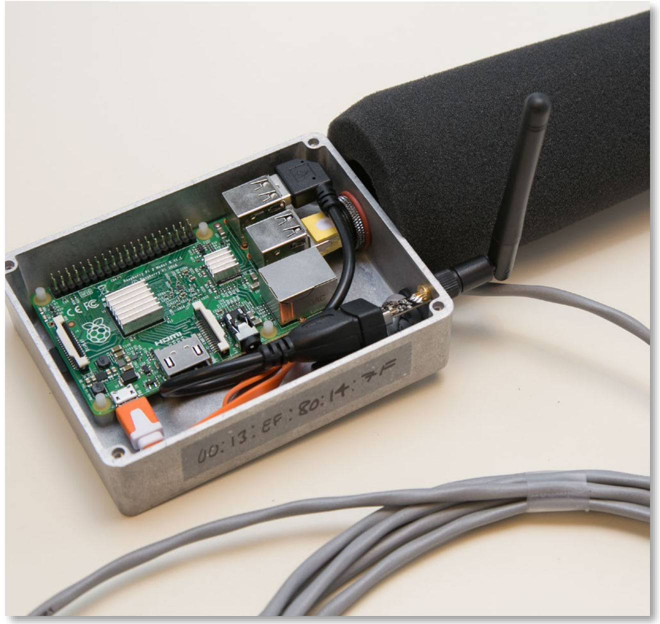

AFRICA
ANTARCTICA


#np 101,702
#love 96,223
#healthcare 93,449
#nowplaying 85,525
#vscocam 84,024
#lrt 74,398
#jesuischarlie 70,950
#askbelieber 69,622
#trndnl 68,686
#selfie 62,228
#melopidoenfnac 61,067
#photo 59,745
#endomondo 59,528
#camfollowme 59,514
#followmecaniff 58,705
#tbt 57,907
#askdirectioner 56,377


180k
150k
120k
90.0k
60.0k
30.0k
0
2h Thu Jan 1 04:00:00 2015
Tue Jan 6 18:00:00 2015
Mon Jan 12 08:00:00 2015
Sat Jan 17 22:00:00 2015

Sensor data

SOUNDS OF NEW YORK CITY



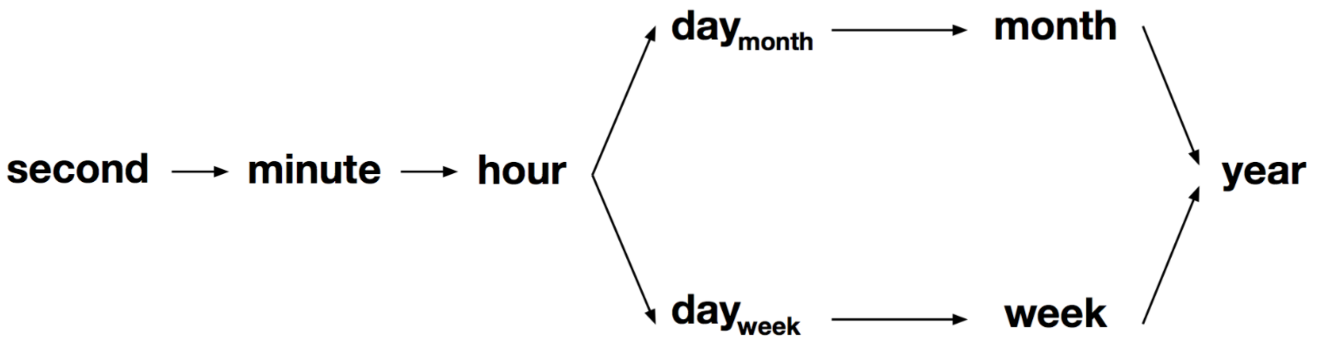
45 sensors
3 boroughs
34 years of high-resolution decibel data (if combined)

Time Lattice

- Support queries having constraints at multiple time resolutions:
  - Average decibel at each hour of the day.
  - Average decibel at day of the week.
  - Average decibel at each day of the week, between 8am – 6pm.
- Support range queries at multiple resolutions:
  - Average decibel between March 1st and March 15th, at hour resolution.
- Support updates from new data
- **Small memory** overhead.
- Allow low latency queries over large time series **(< 500 ms)**.

Time Lattice

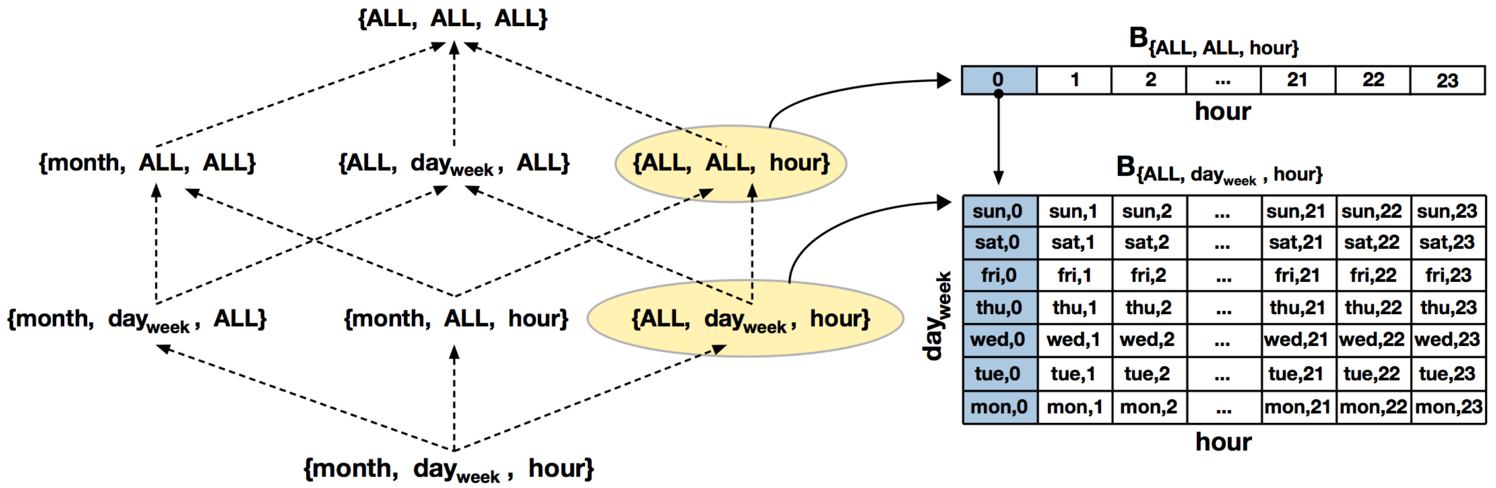
- Data structure that supports multiple resolution queries at interactive rates.
- Makes use of the implicit hierarchy present in temporal resolutions to materialize a sub-lattice of a data cube.



[Miranda et al., 2019]

Time Lattice

- Data structure that supports multiple resolution queries at interactive rates.
- Makes use of the implicit hierarchy present in temporal resolutions to materialize a sub-lattice of a data cube.

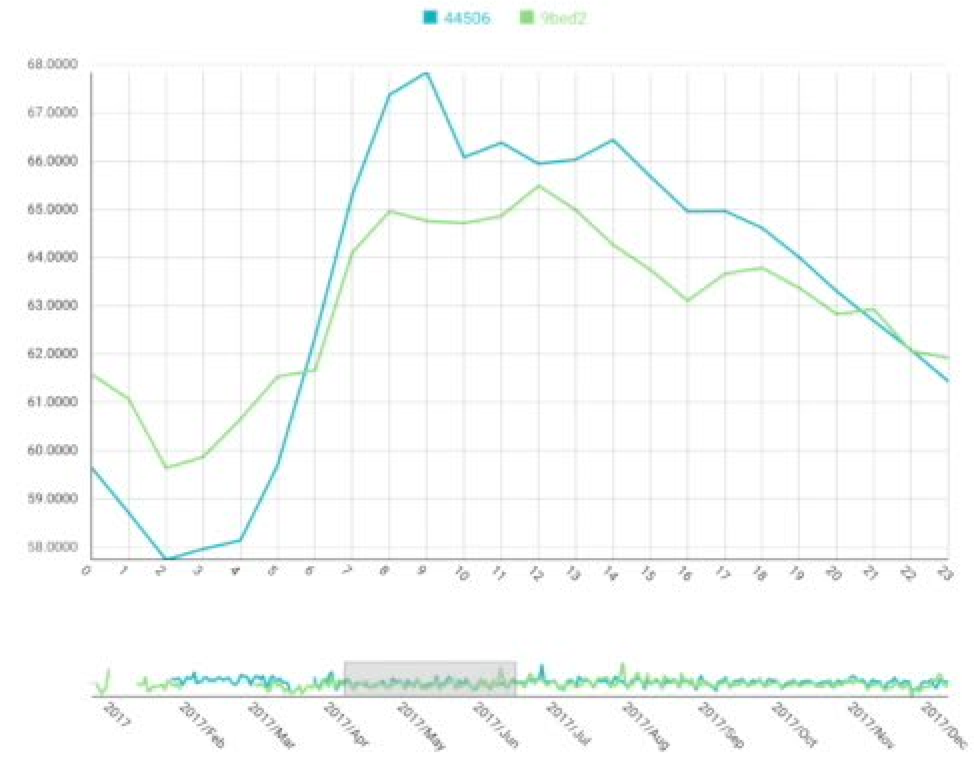


[Miranda et al., 2019]

Time Lattice

Card Config

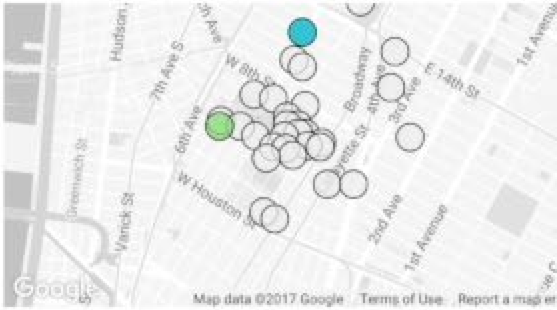
Default title



Card title: Default title

Card width: 750

Card height: 650



Start date/time: 2016-10-01 00:00

End date/time: 2017-12-31 23:59

Constraints (Months)

JAN FEB MAR APR MAY JUN

JUL AUG SEP OCT NOV DEC

Constraints (Week Days)

MON TUE WED THU FRI SAT SUN

Constraints (Hours)

0 1 2 3 4 5 6 7 8 9 10 11

12 13 14 15 16 17 18 19 20 21 22 23

Group by: Hour

Resolution: Default

Aggregation: Average

Deviation scale: 0

Percentile: 0.9

Min value: 0

Max Value: 10

+