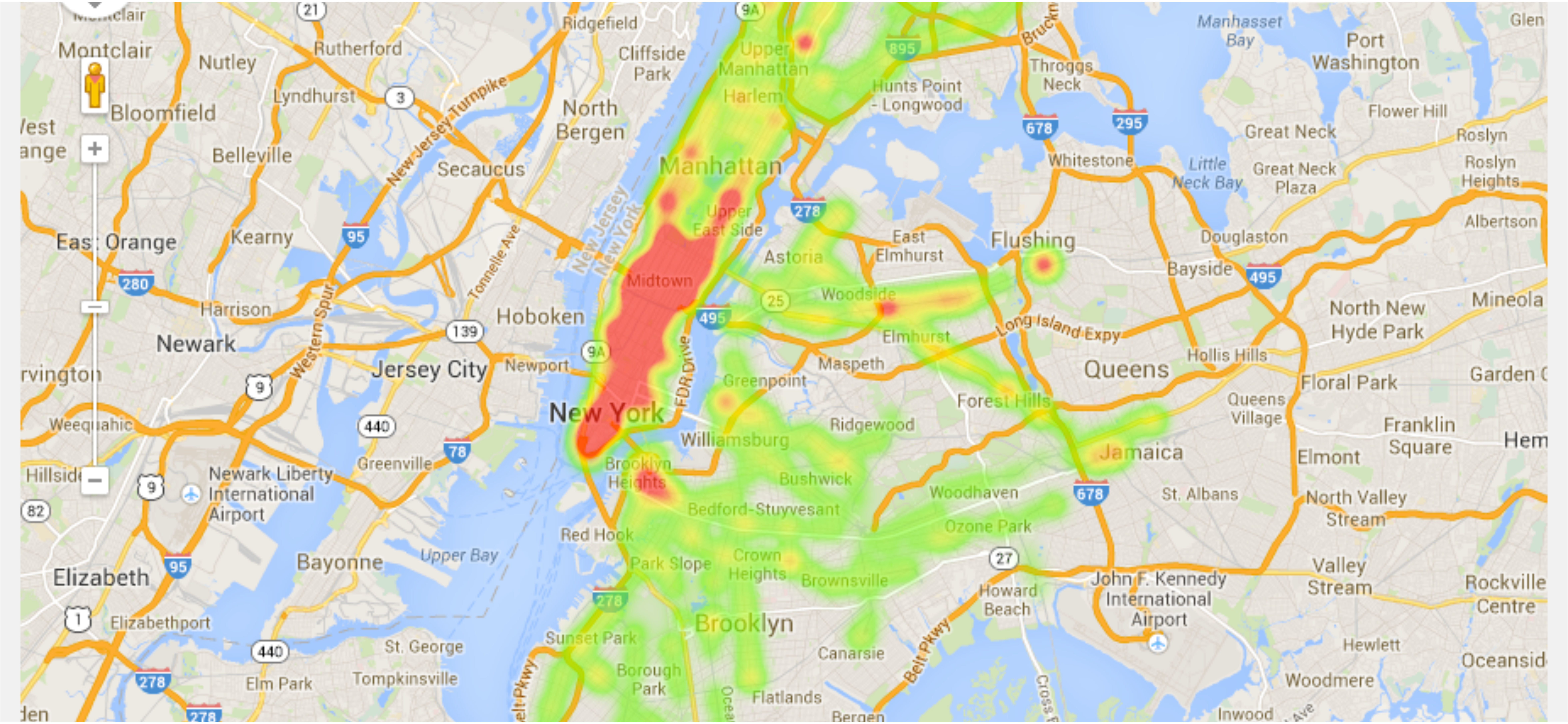
Advanced Data Management (CSCI 640/490)

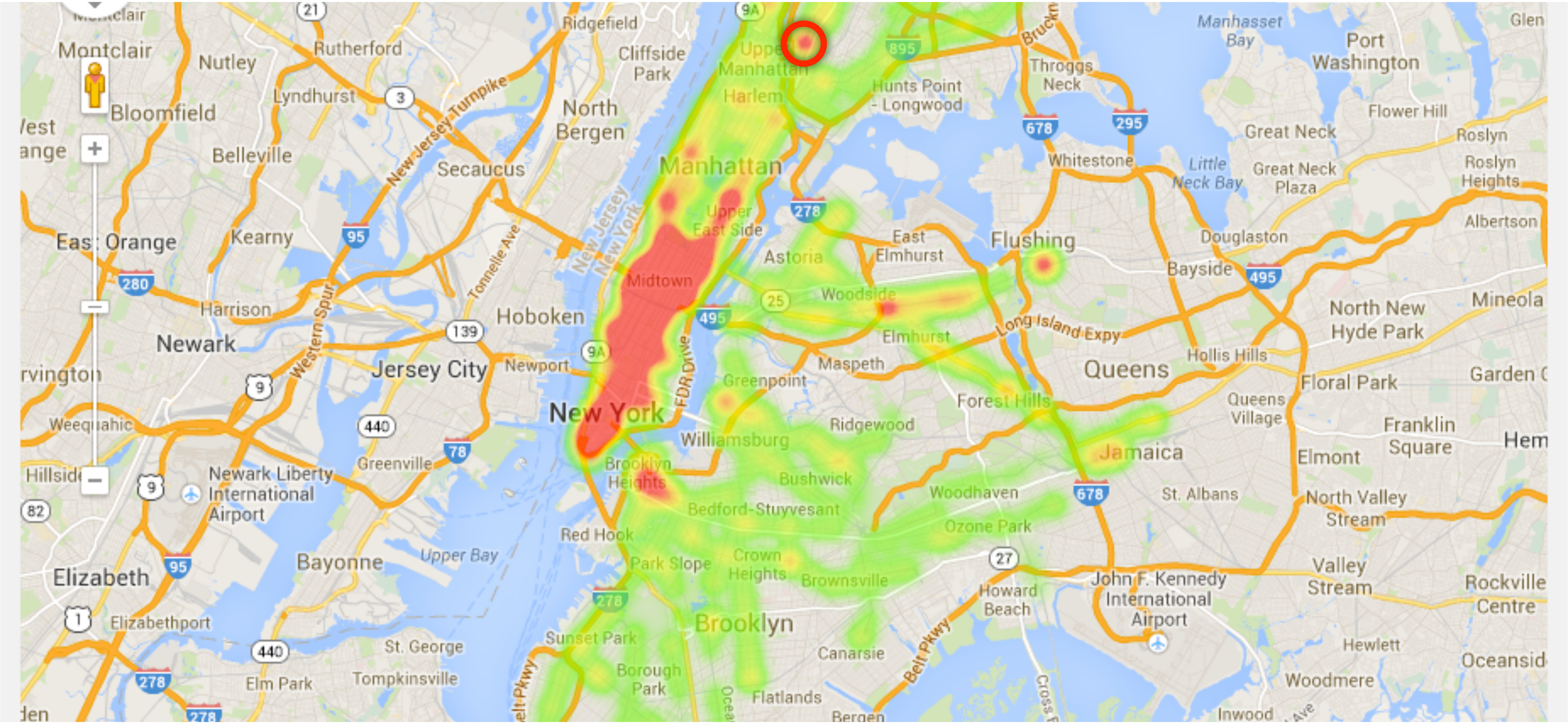
Spatial Data

Dr. David Koop

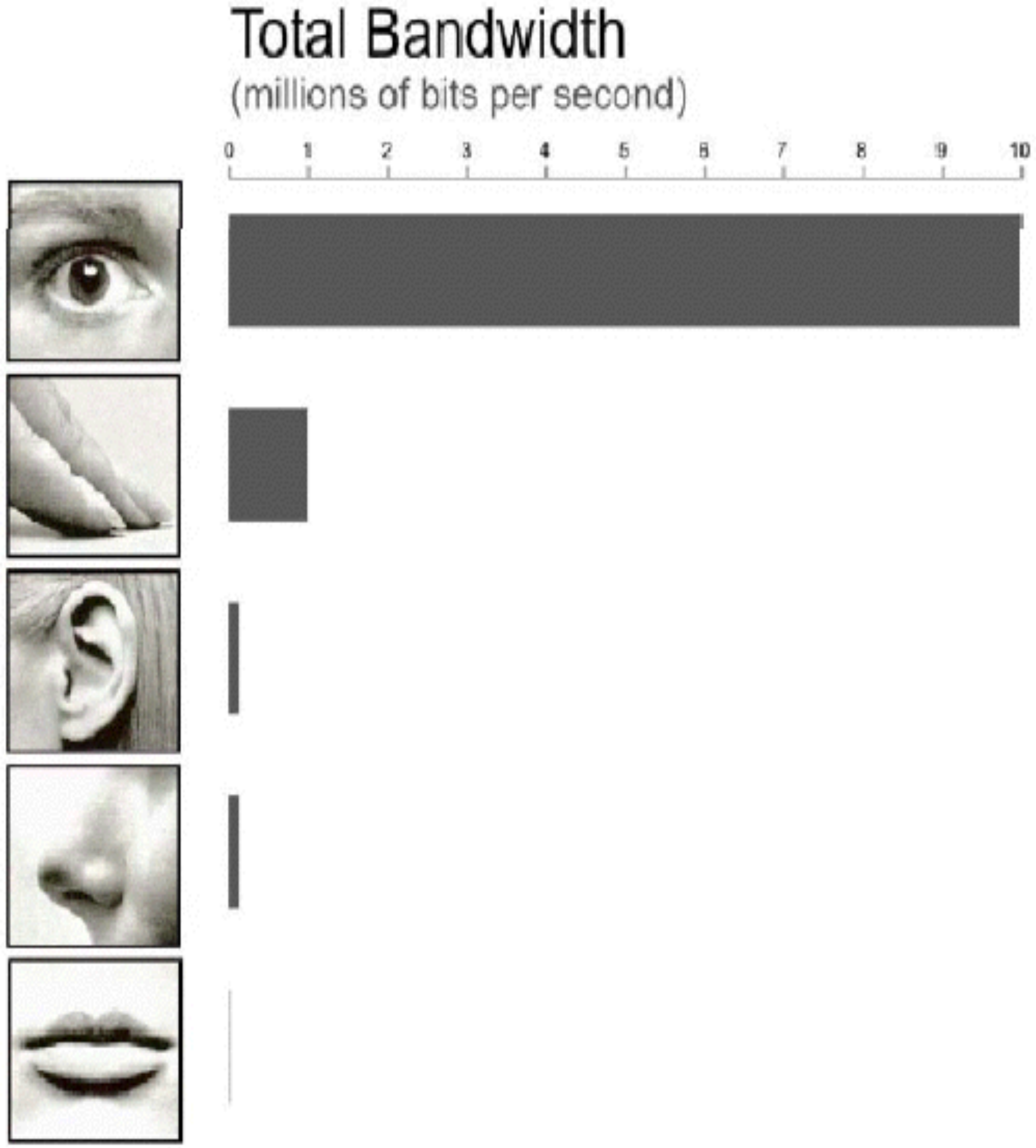
Data Exploration Through Visualization



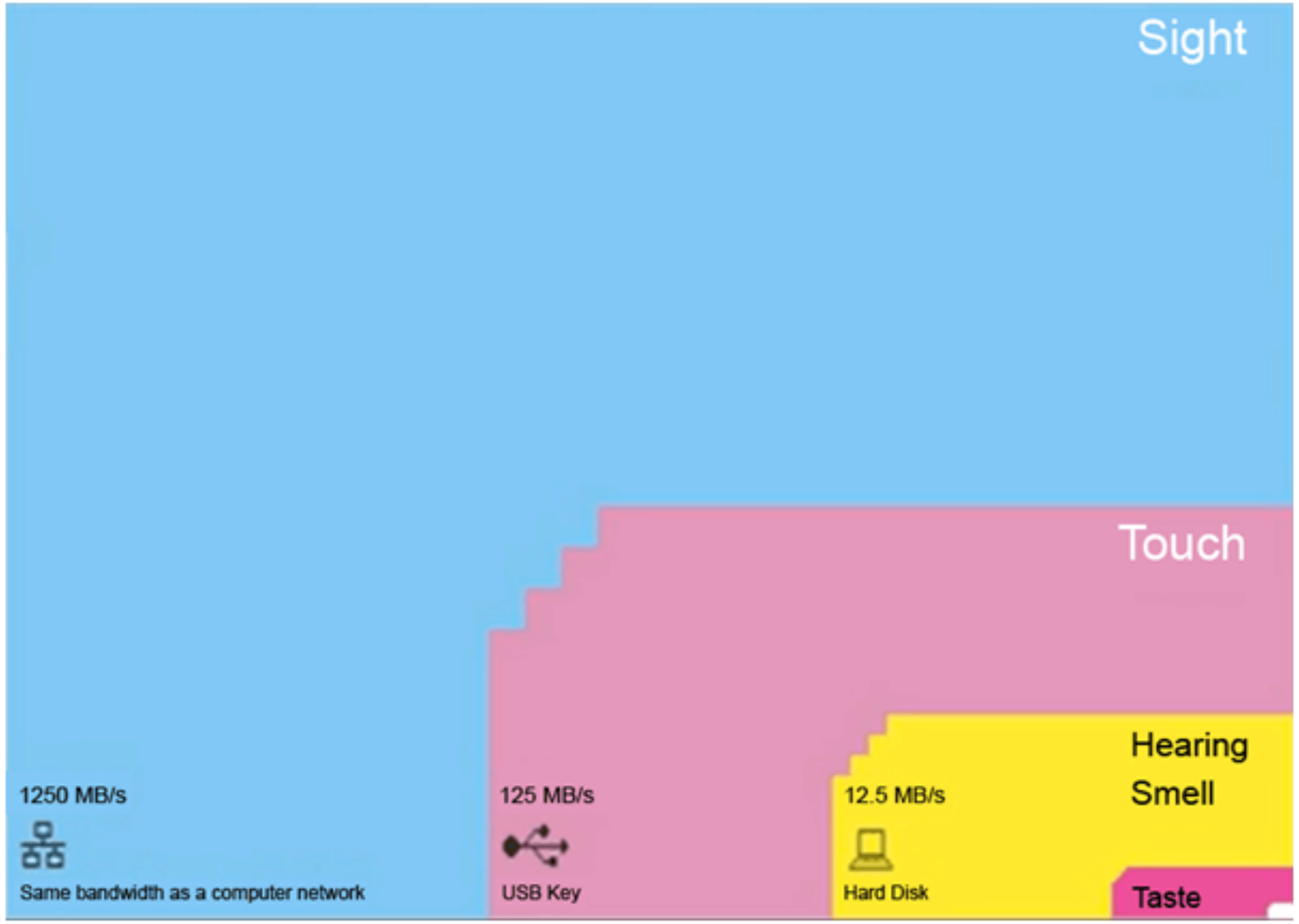
Data Exploration Through Visualization



Why do we visualize data?



[via A. Lex]



[T. Nørretranders]

Why Visual?

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

[F. J. Anscombe]

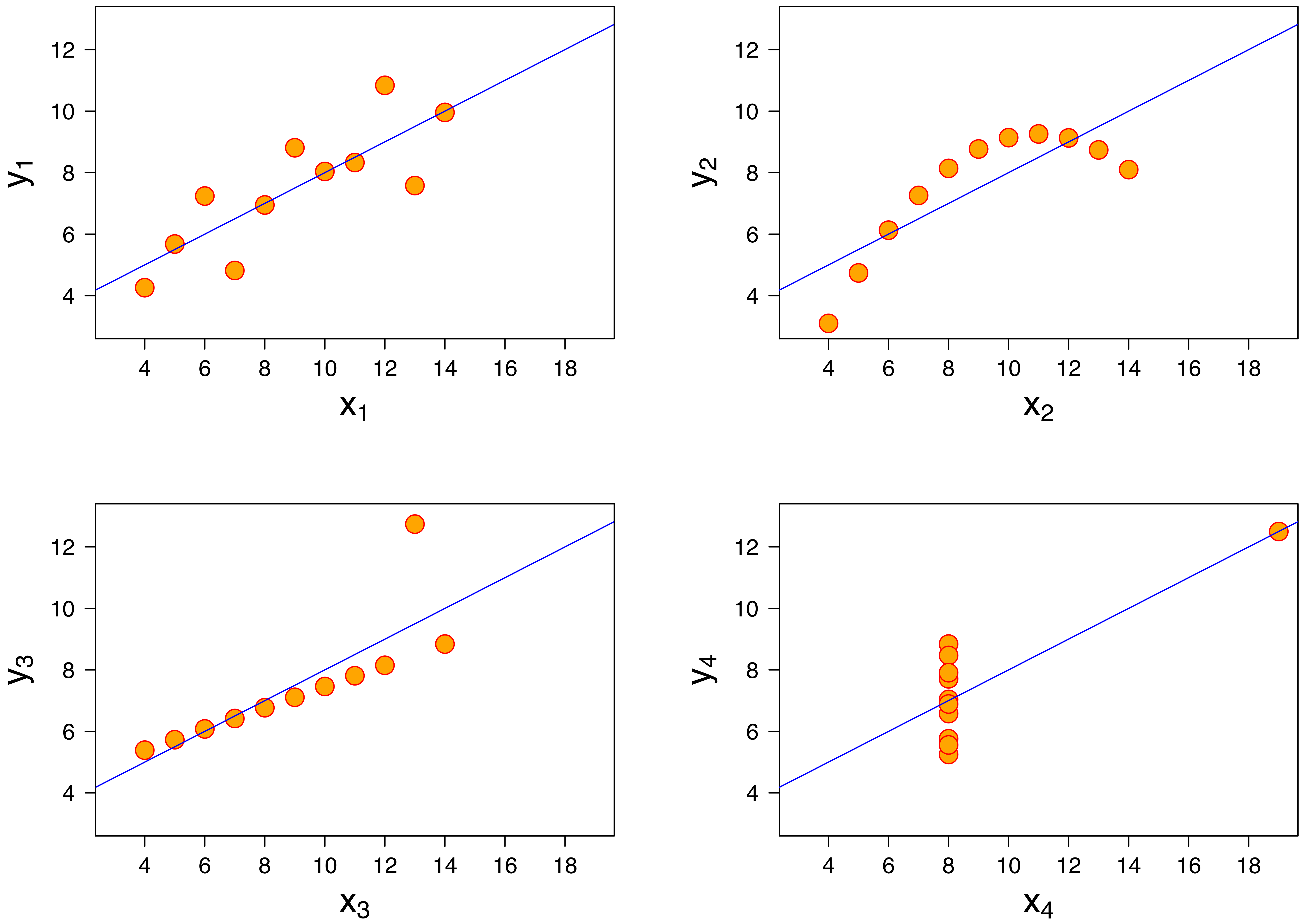
Why Visual?

I		II		III		IV	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816

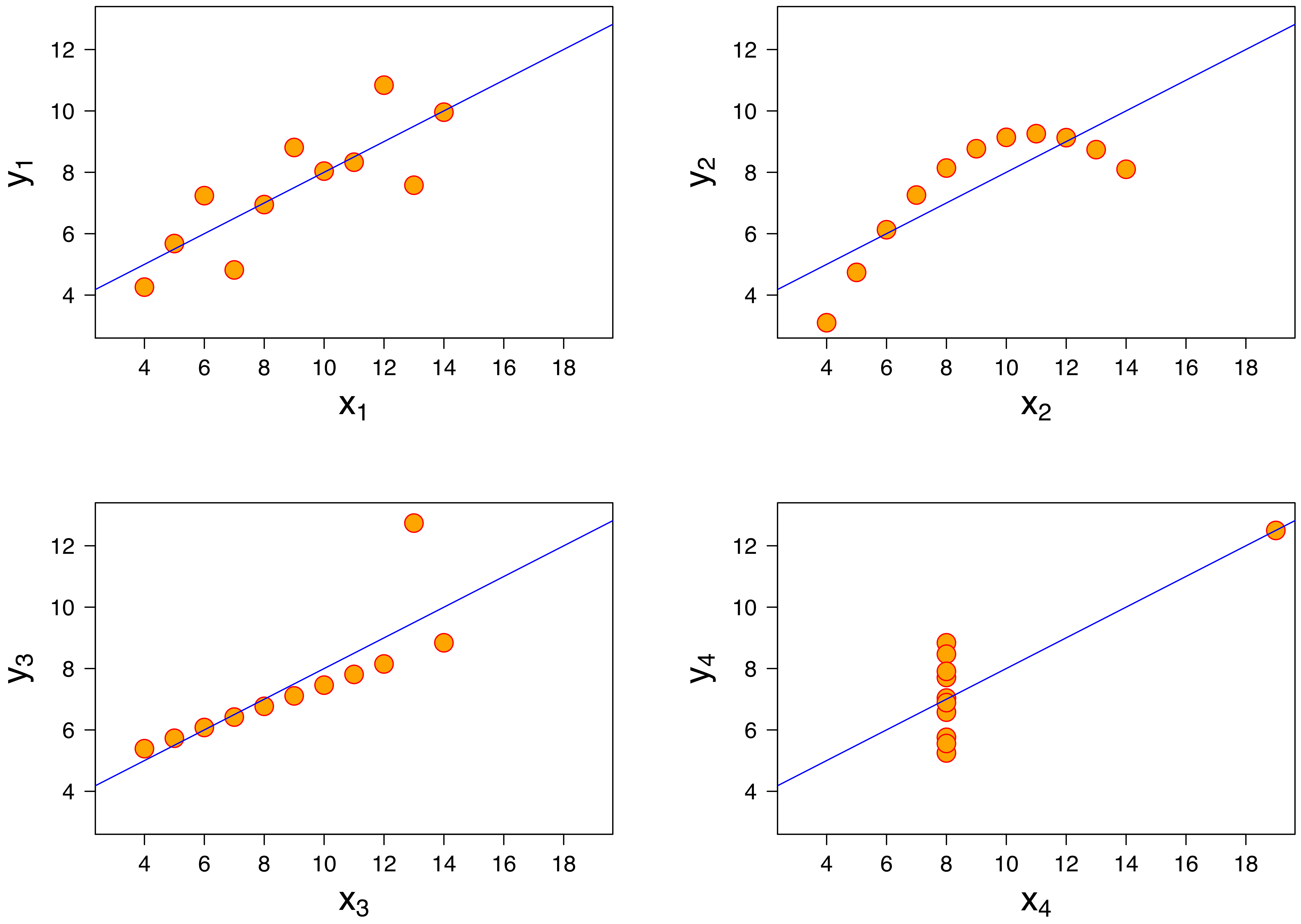
[F. J. Anscombe]

Why Visual?



[F. J. Anscombe]

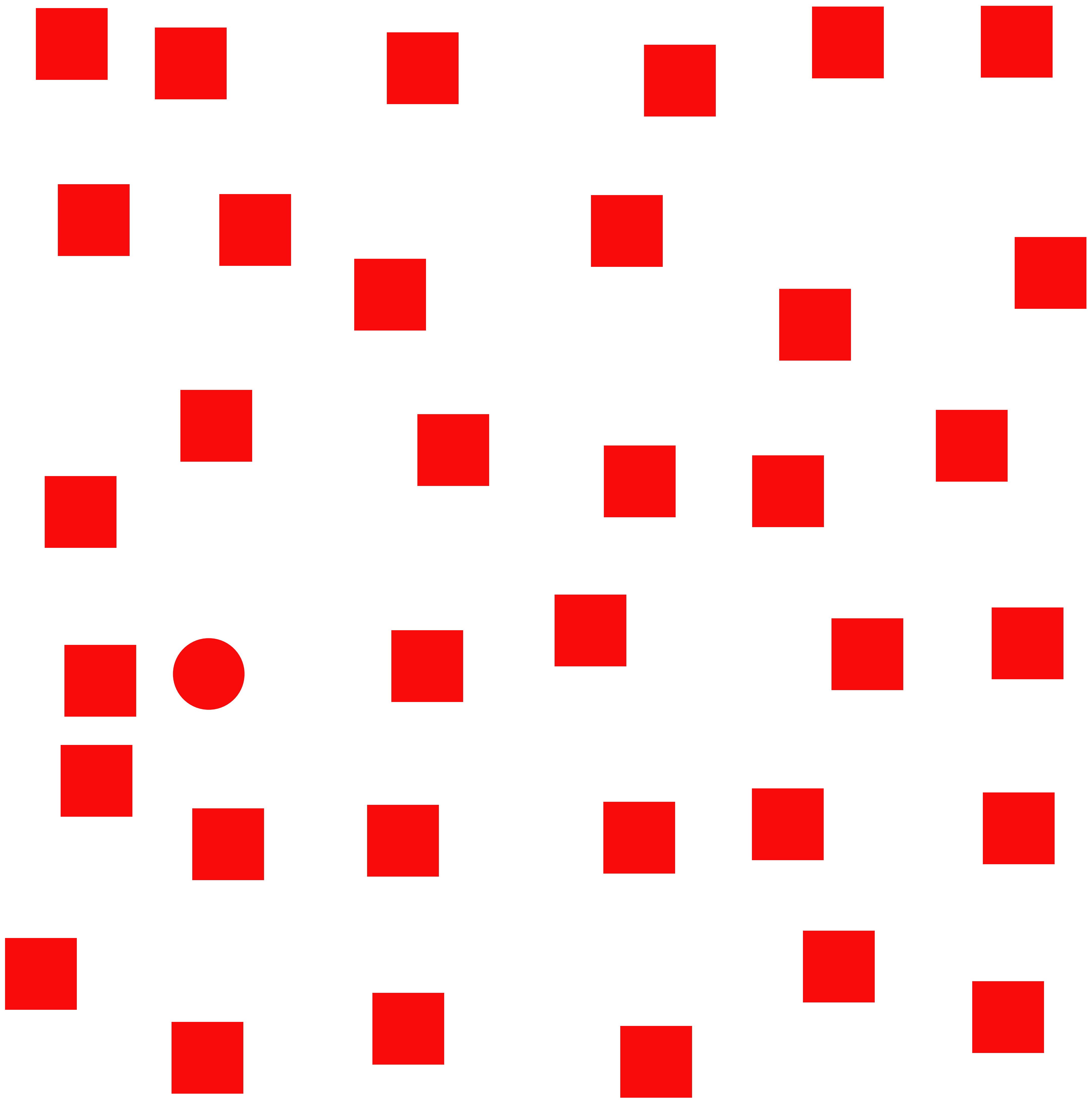
Why Visual?



Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816

[F. J. Anscombe]

Visual Pop-out



[C. G. Healey]

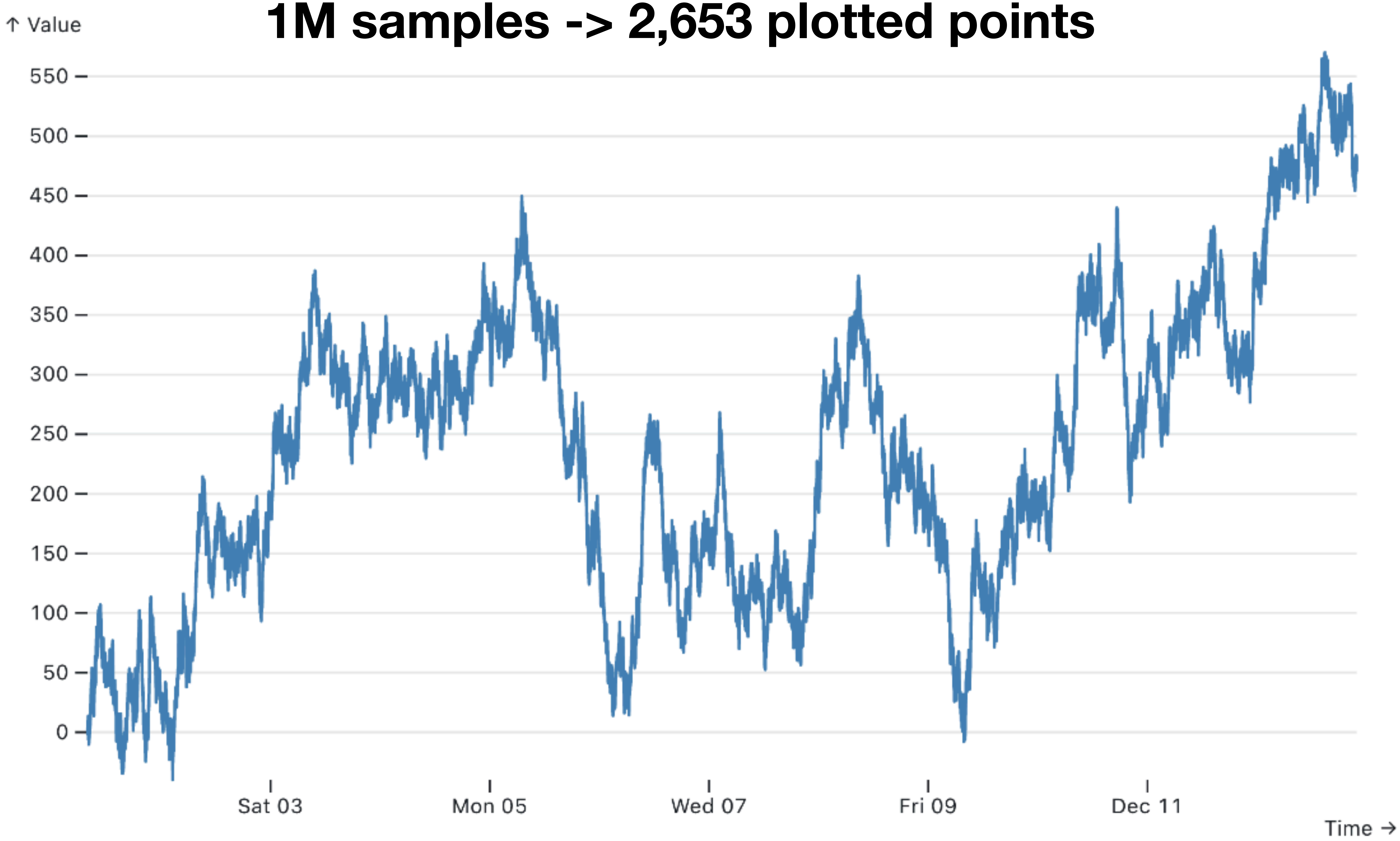
Supporting Scalable Visualization

- Two Problems:
  - **Lots of data**, how to display (encode) it
  - User **interaction** is key to gaining insight, requires **low latency**
- Addressing big data:
  - Encoding should focus on **available resolution**, not size of data
  - Approaches:
    - Sampling
    - Modeling
    - **Binning**
  - Bin → Aggregate (→ Smooth) → Plot

[J. Heer]

Time Series Aggregation

1M samples -> 2,653 plotted points



[J. Heer]

Time Series Aggregation

- Insight: the **resolution** is bound by the **number of pixels**
- Compute average value per pixel (1 point/pixel)
  - …this may miss extreme (min, max) values
- Plot min/max values per pixel (2 points/pixel)
  - …this does better, but still misrepresents
- M4: min/max values & timestamps (4 points/pixel)
  - …this provides provable fidelity to the full data!

[Jugel, 2014, via J. Heer]

Effects of Latency

- **Higher latency** leads to...
  - Reduced user activity and data set coverage
  - Significantly fewer brushing actions
  - Less observation, generalization & hypothesis
- **Interaction effect**: Exposure to delay reduces subsequent performance in low-latency interface.
- Different interactions exhibit **varied sensitivity** to latency. Brushing is highly sensitive!

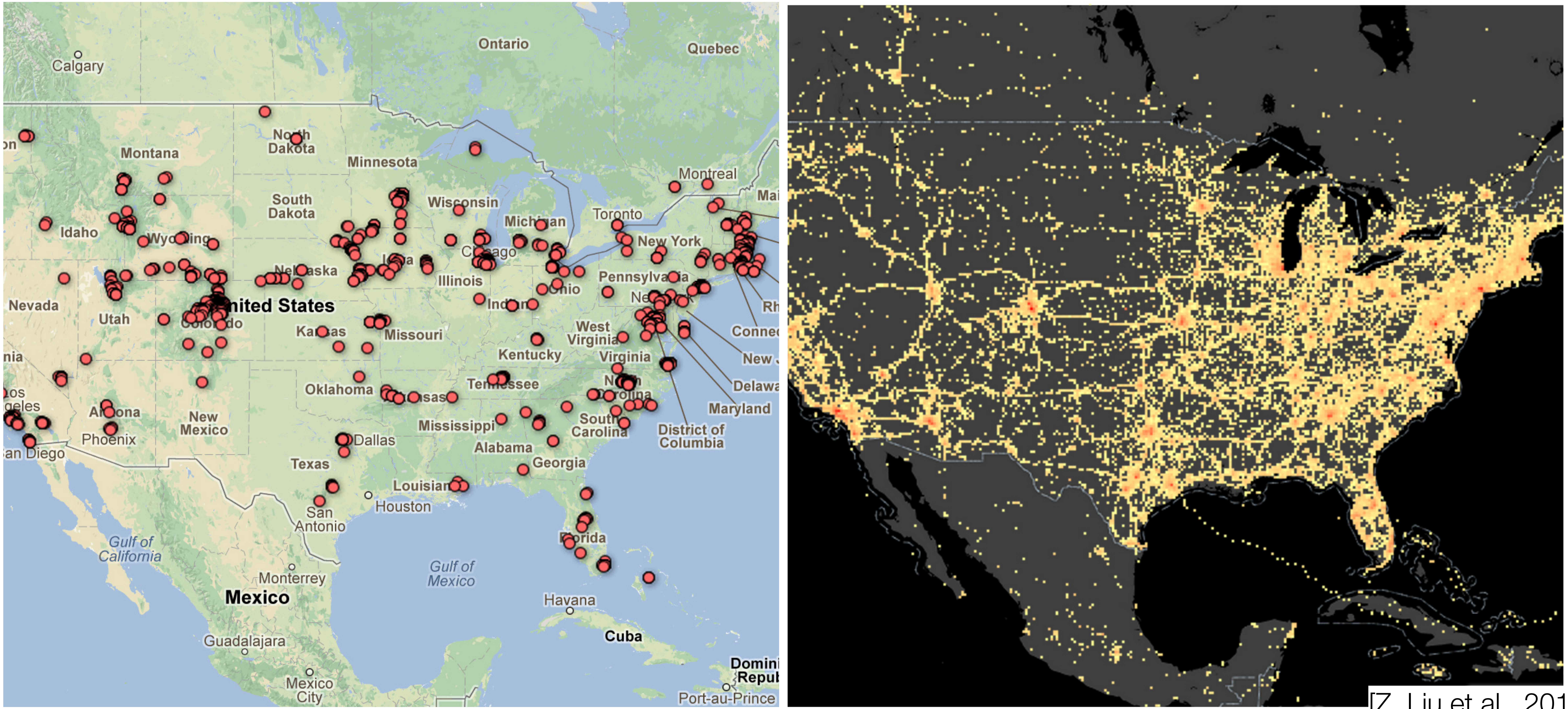
[Liu et al. via J. Heer]

Interactive Scalability Solutions

- Use Database Technology: databases built for scalability
- Client-side Indexing (Data Cubes): take advantage of data structure
- Prefetching: load data before requests based on predictions
- Approximation: show estimates early but with error information

[J. Heer]

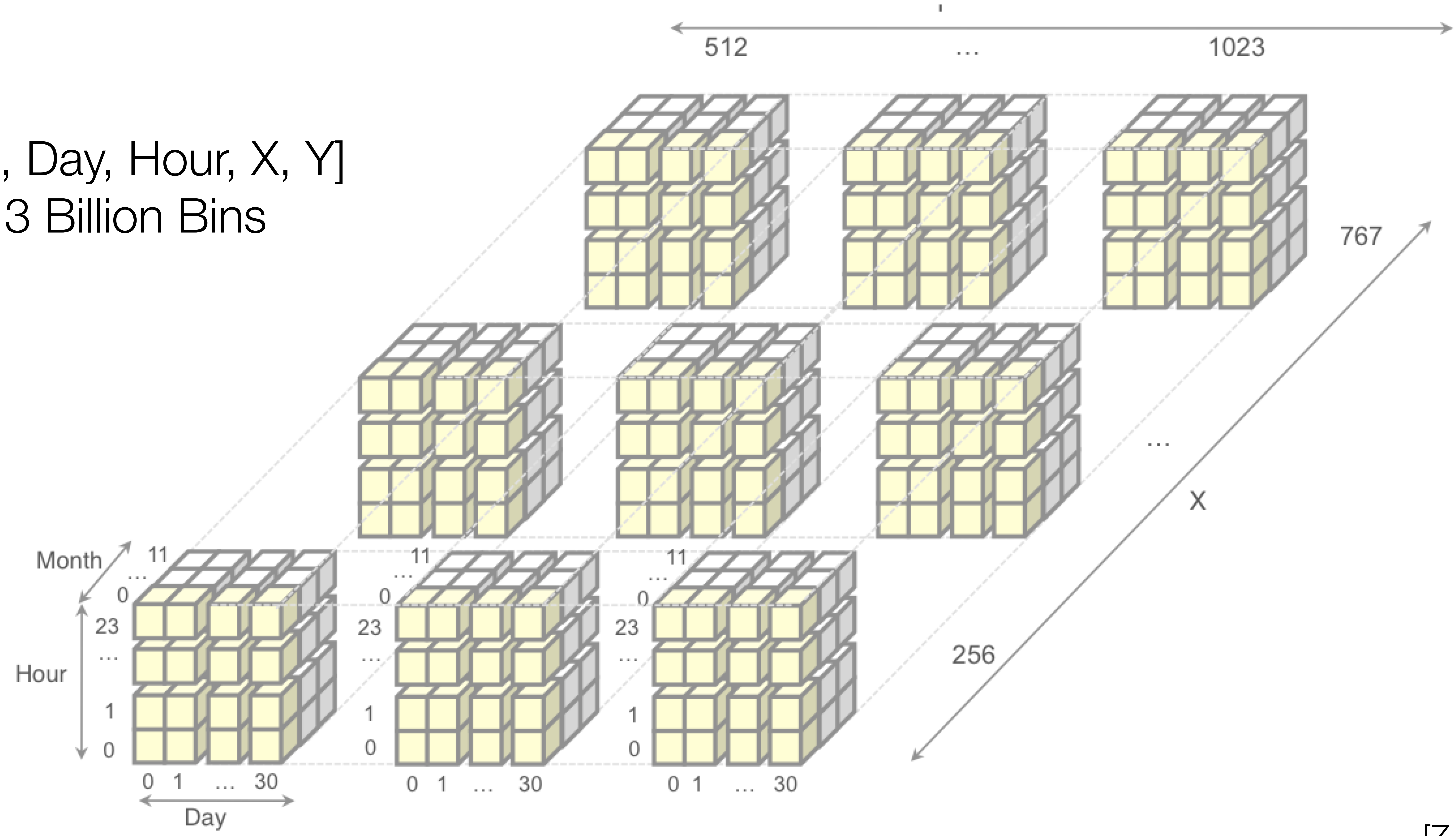
Sampling vs. Aggregation



[Z. Liu et al., 2013]

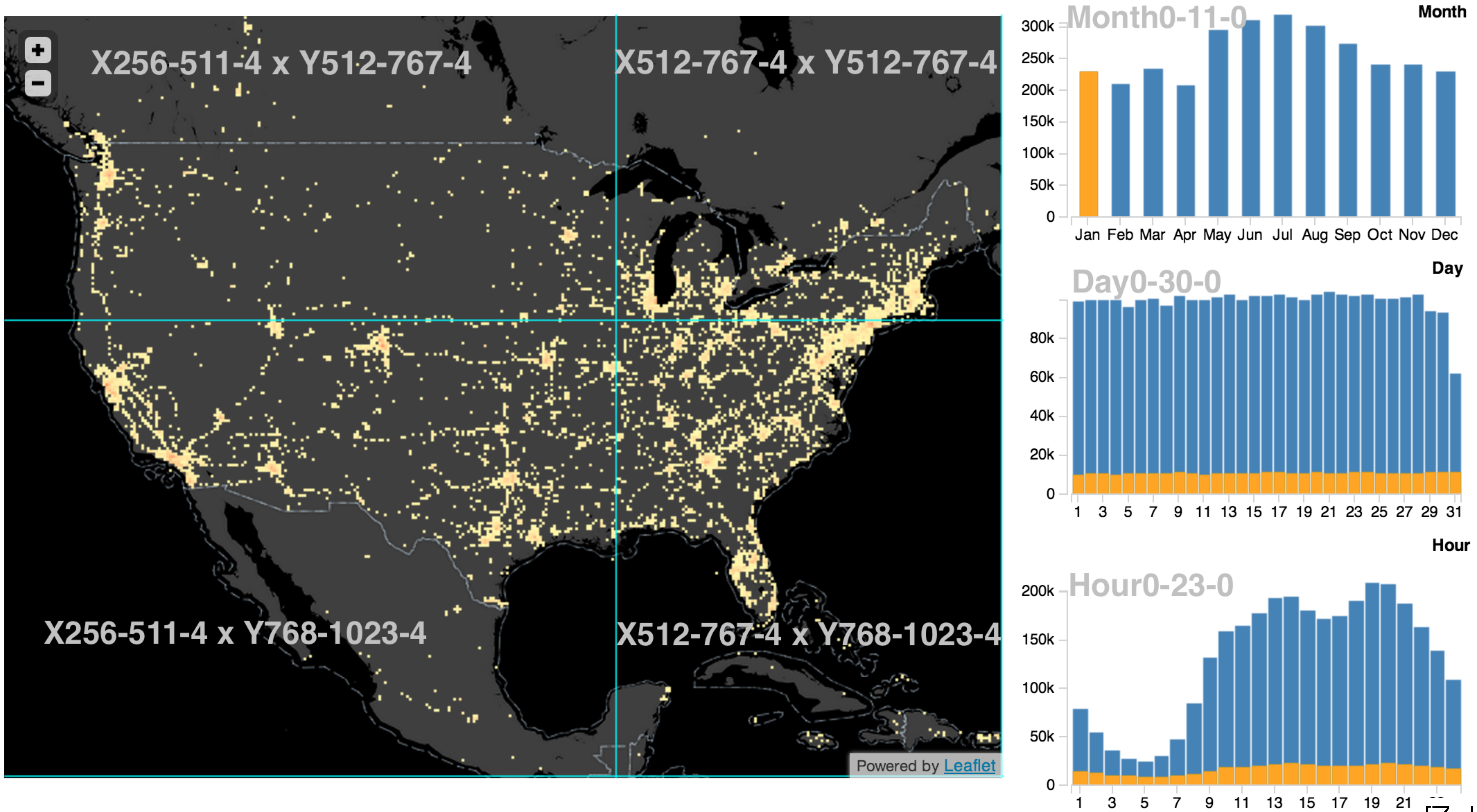
Full 5-D Data Cube

[Month, Day, Hour, X, Y]
~2.3 Billion Bins



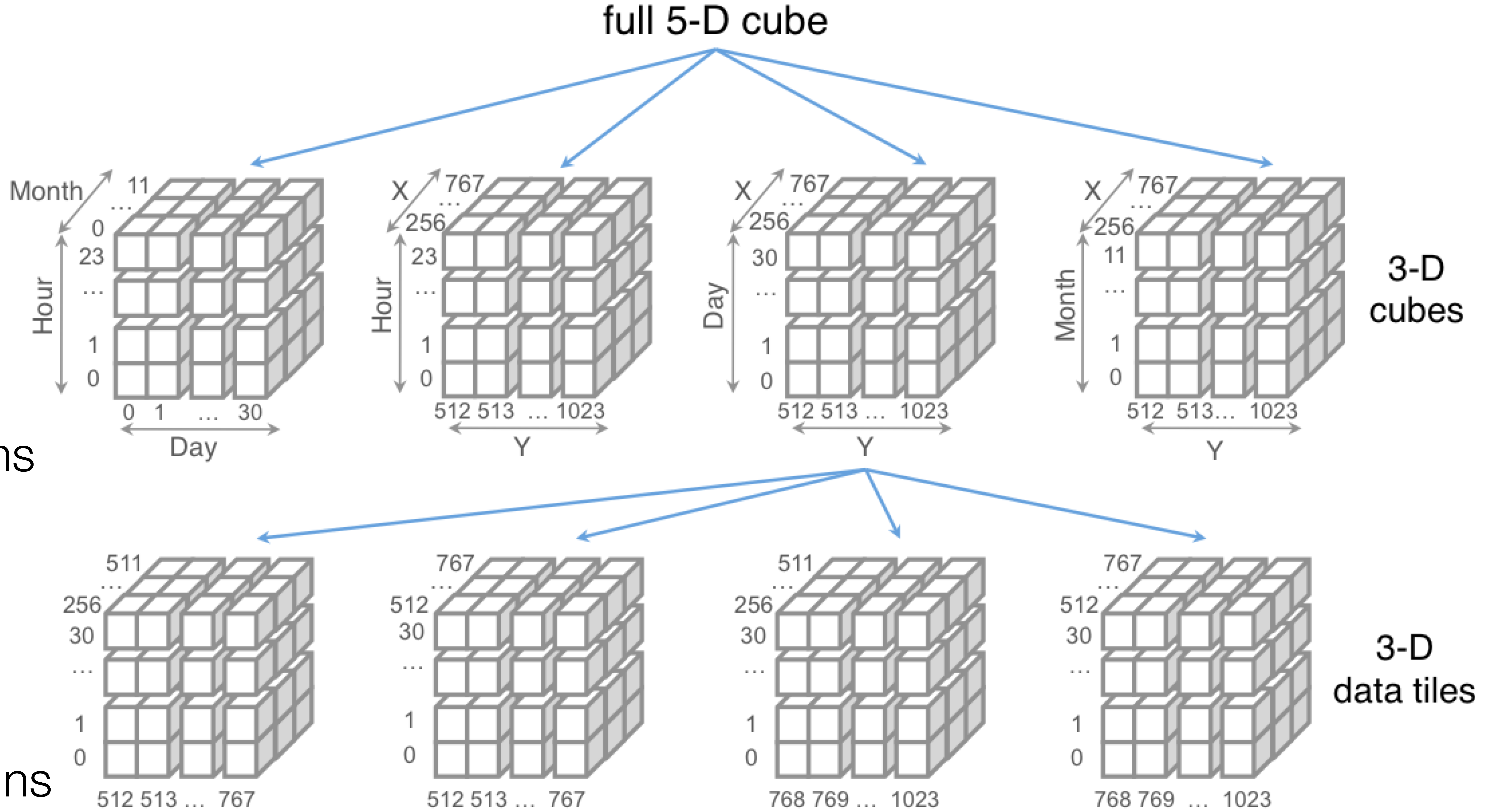
[Z. Liu et al., 2013]

Break into Tiles



[Z. Liu et al., 2013]

Data Cube Decomposition

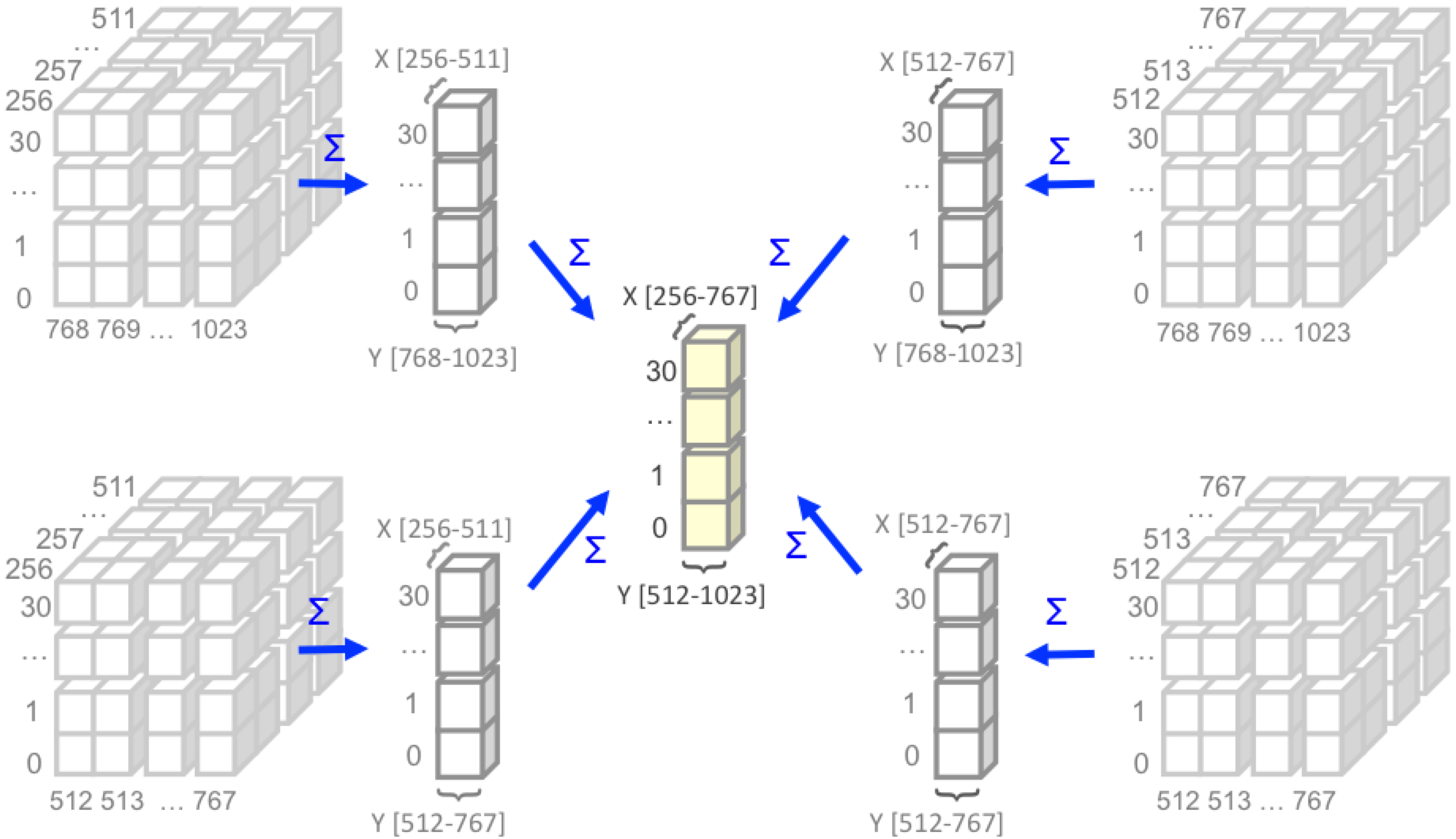


~2.3 B Bins

~17.6 M Bins

[Z. Liu et al., 2013]

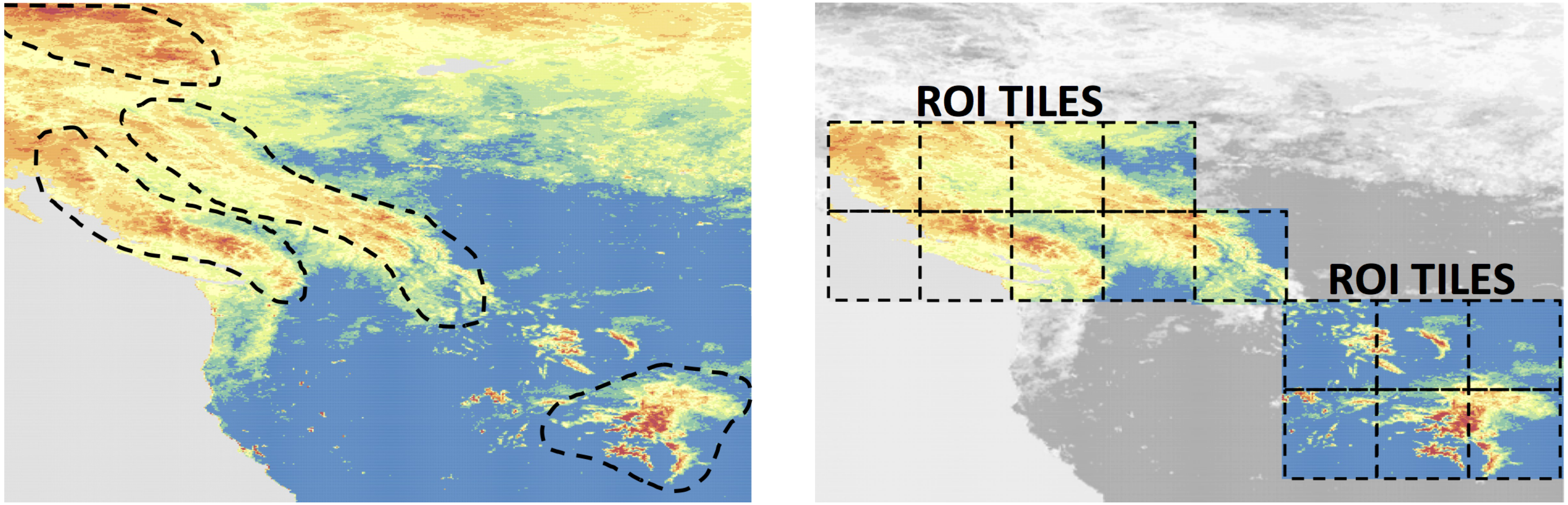
Aggregation of Tiles for Results



[Z. Liu et al., 2013]

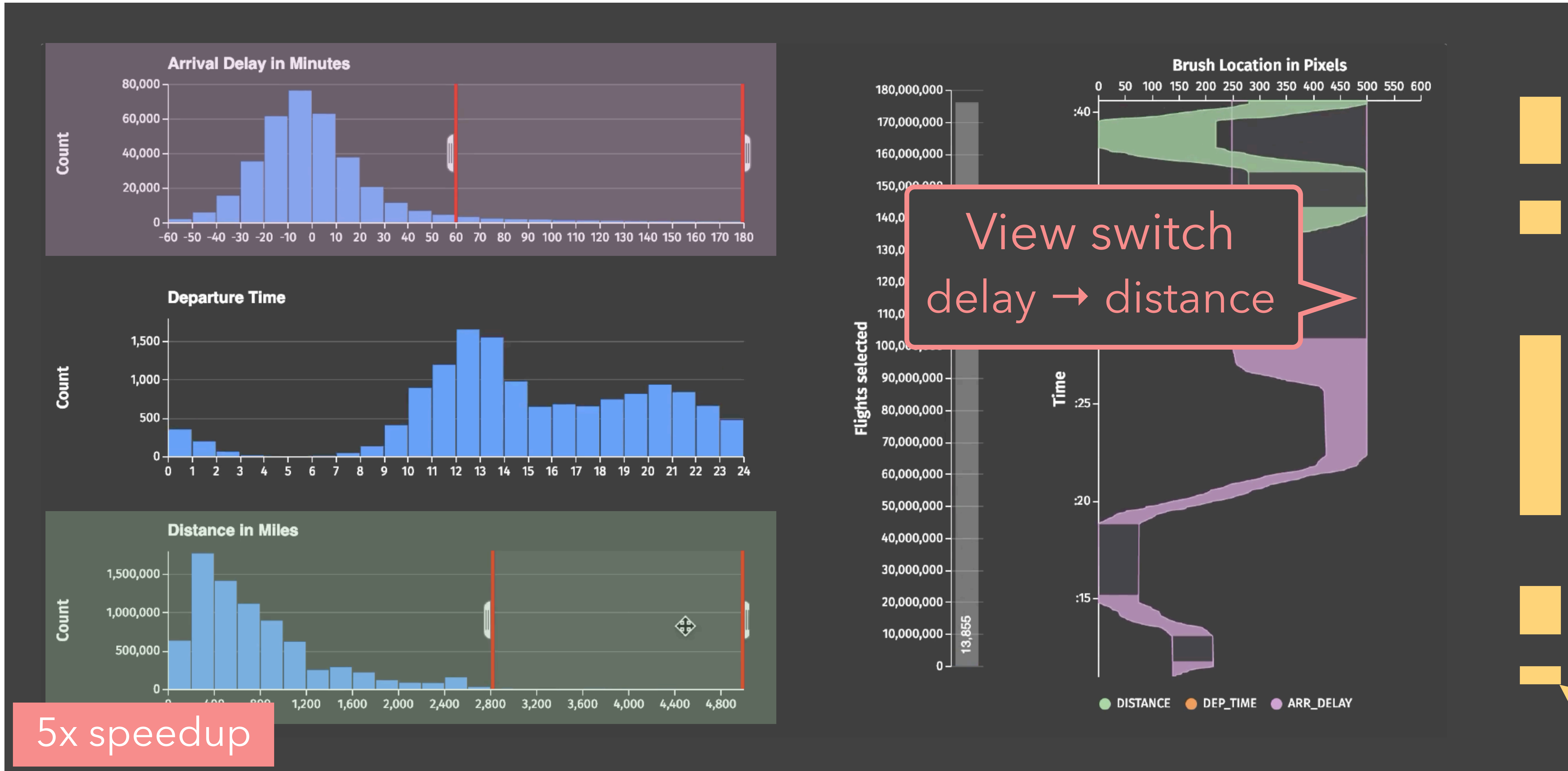
Prefetching

- Predict which tiles a user will need next and prefetch those
  - Use common patterns (zoom, pan)
  - Use regions of interest (ROIs)



[Battle et al., 2016]

Latency Differences in Tasks



View switch delay → distance

5x speedup

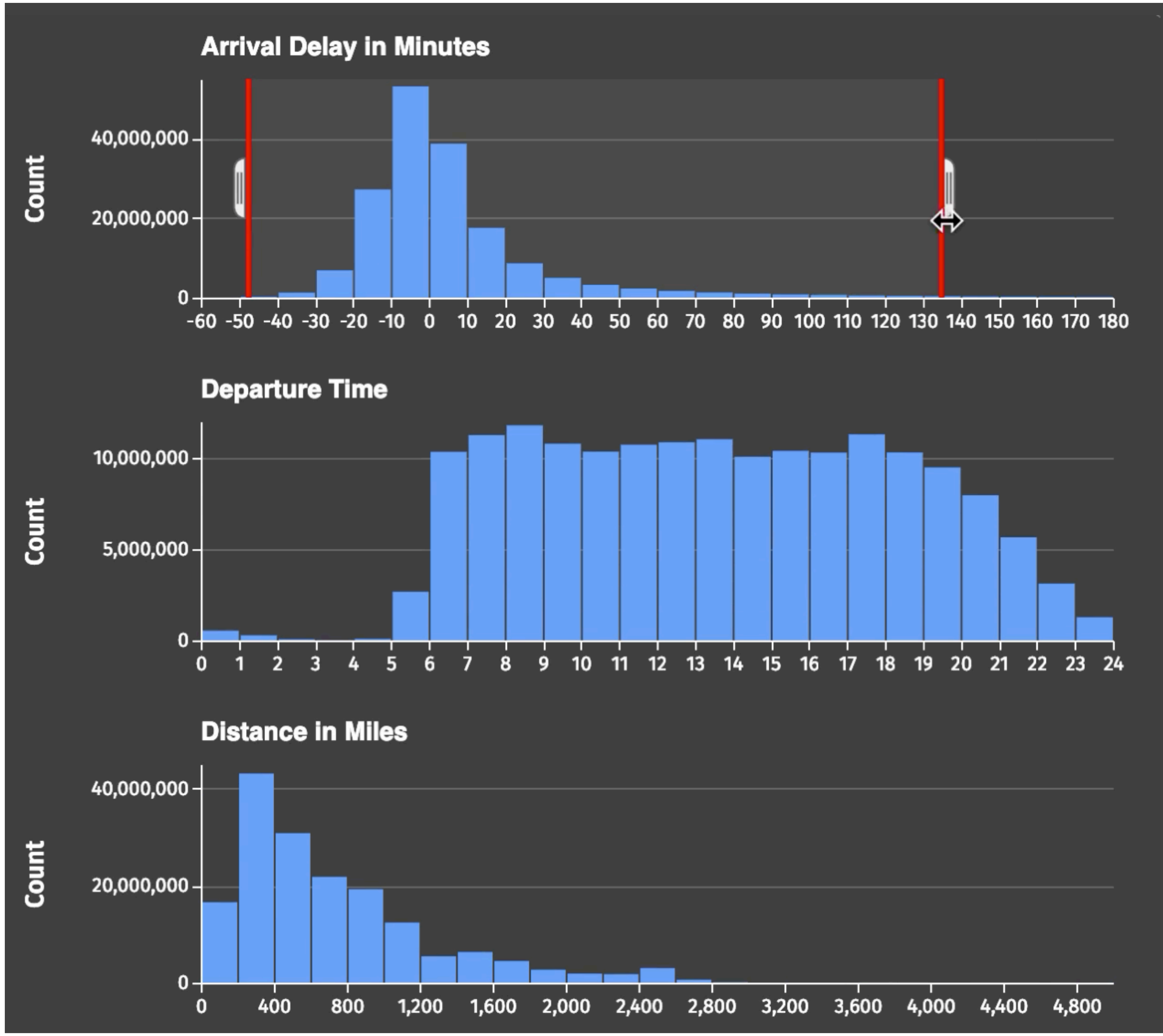
Brushing interactions

👁 Brushing is more common and people are sensitive to latencies.

💡 Prioritize brushing latency over view switching latency.

[D. Moritz et al. via J. Heer]

Task-Prioritized Prefetching



brushes in the precomputed view

serves requests from a data cube

Data Cube. Gray et al. *1997.*

interacts with a new view

query for new data cubes

[D. Moritz et al. via J. Heer]

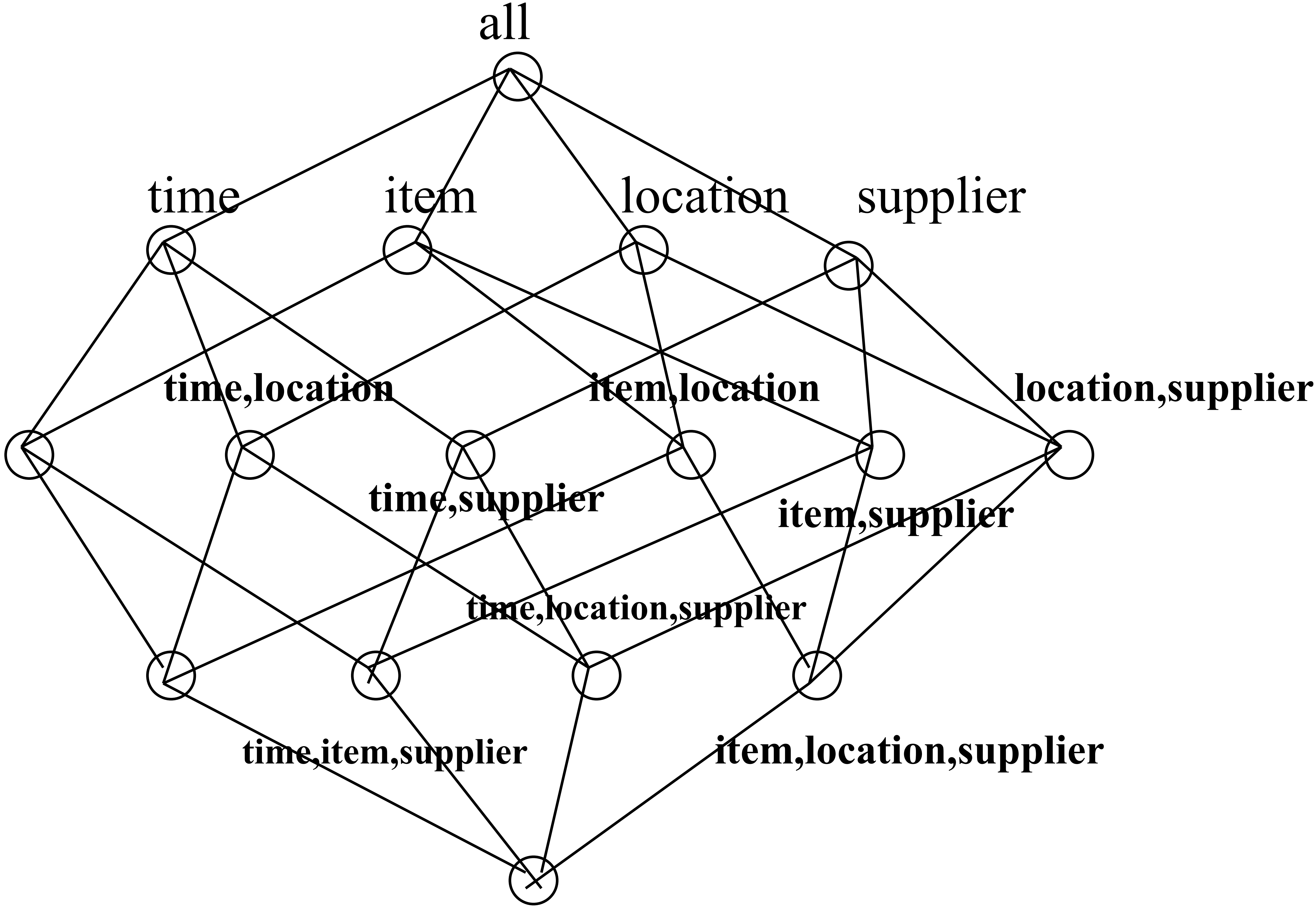
Assignment 5

- Spatial, Graph, and Temporal Data Processing

Data Cubes

J. Han, M. Kamber, and J. Pei

Data Cube: A Lattice of Cuboids



0−D (*apex*) cuboid

1−D cuboids

2−D cuboids

3−D cuboids

4−D (*base*) cuboid

[Han et al., 2011]

Cube Operations

- Roll-up: aggregate up the given hierarchy
- Drill-down: refine down the given hierarchy
- Roll-up and drill-down are "inverses"

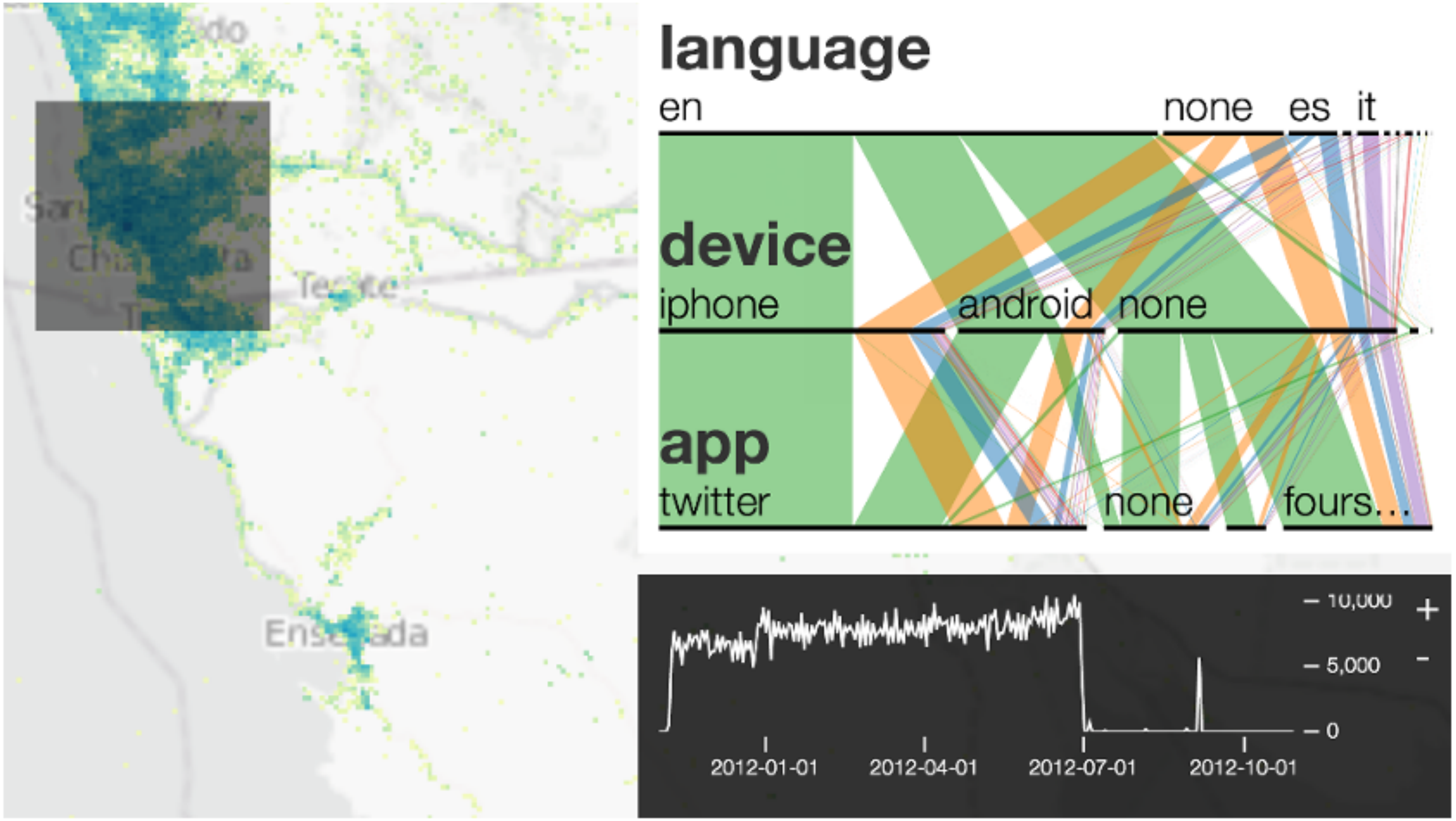
Spatial Data Exploration Motivation

L. Battle

Nanocubes for Real-Time Exploration of Spatiotemporal Datasets

L. Lins, J. T. Klosowski, and C. Scheidegger

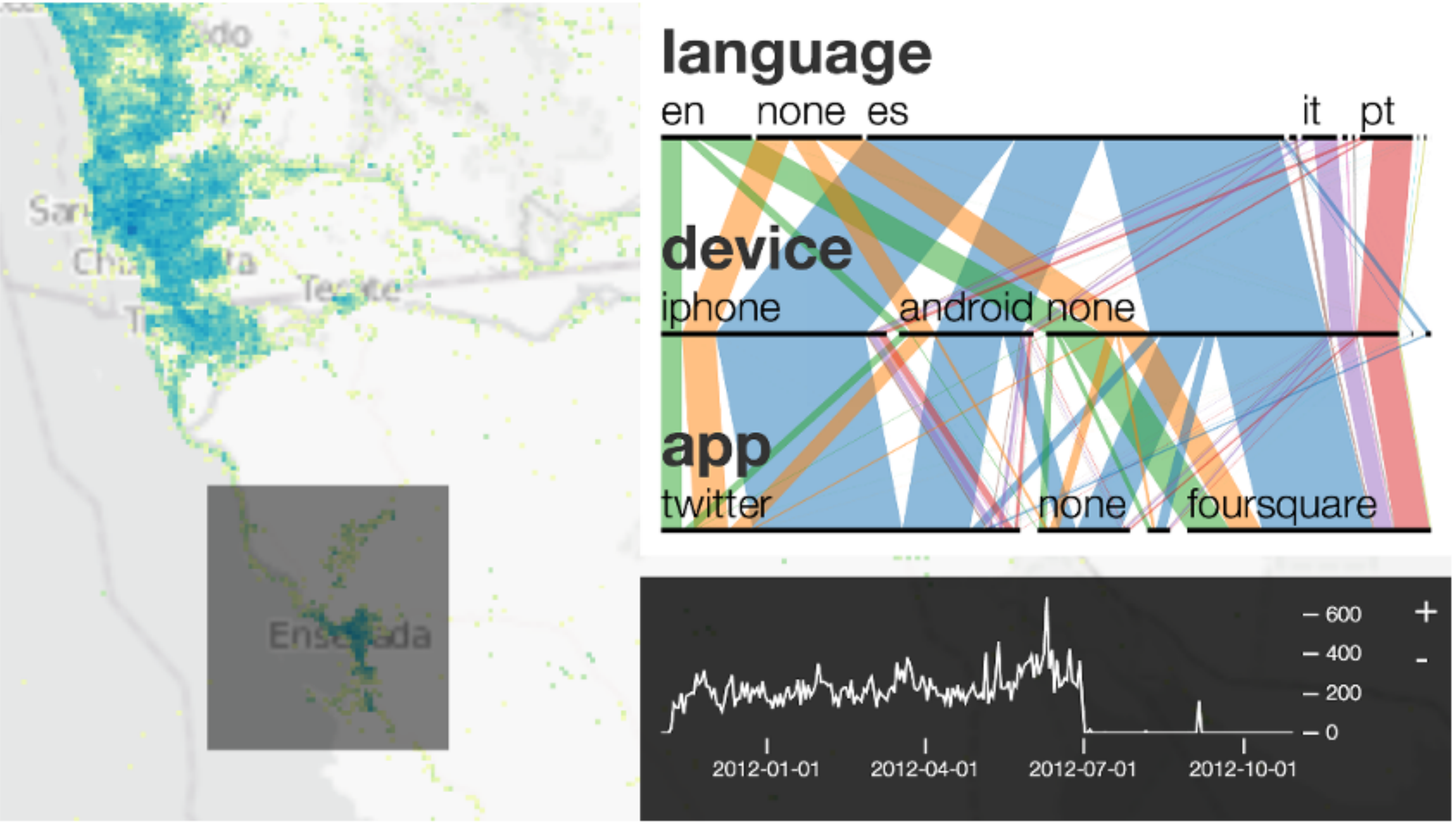
Goal: Interactive Exploration of Data Cubes



Linked view of tweets in San Diego, US

[Lins et. al, 2013]

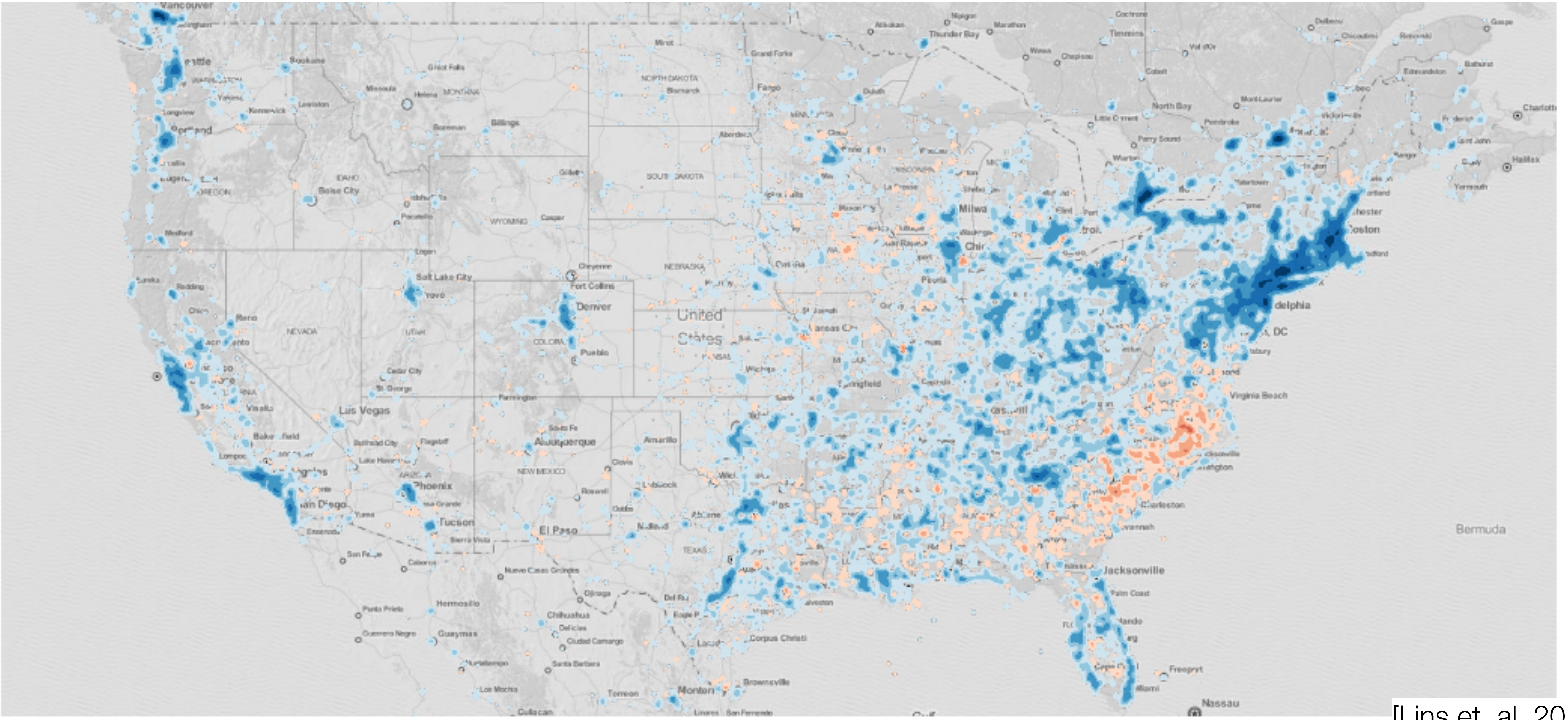
Move to Another Location



Linked view of tweets in Ensenada, Mexico

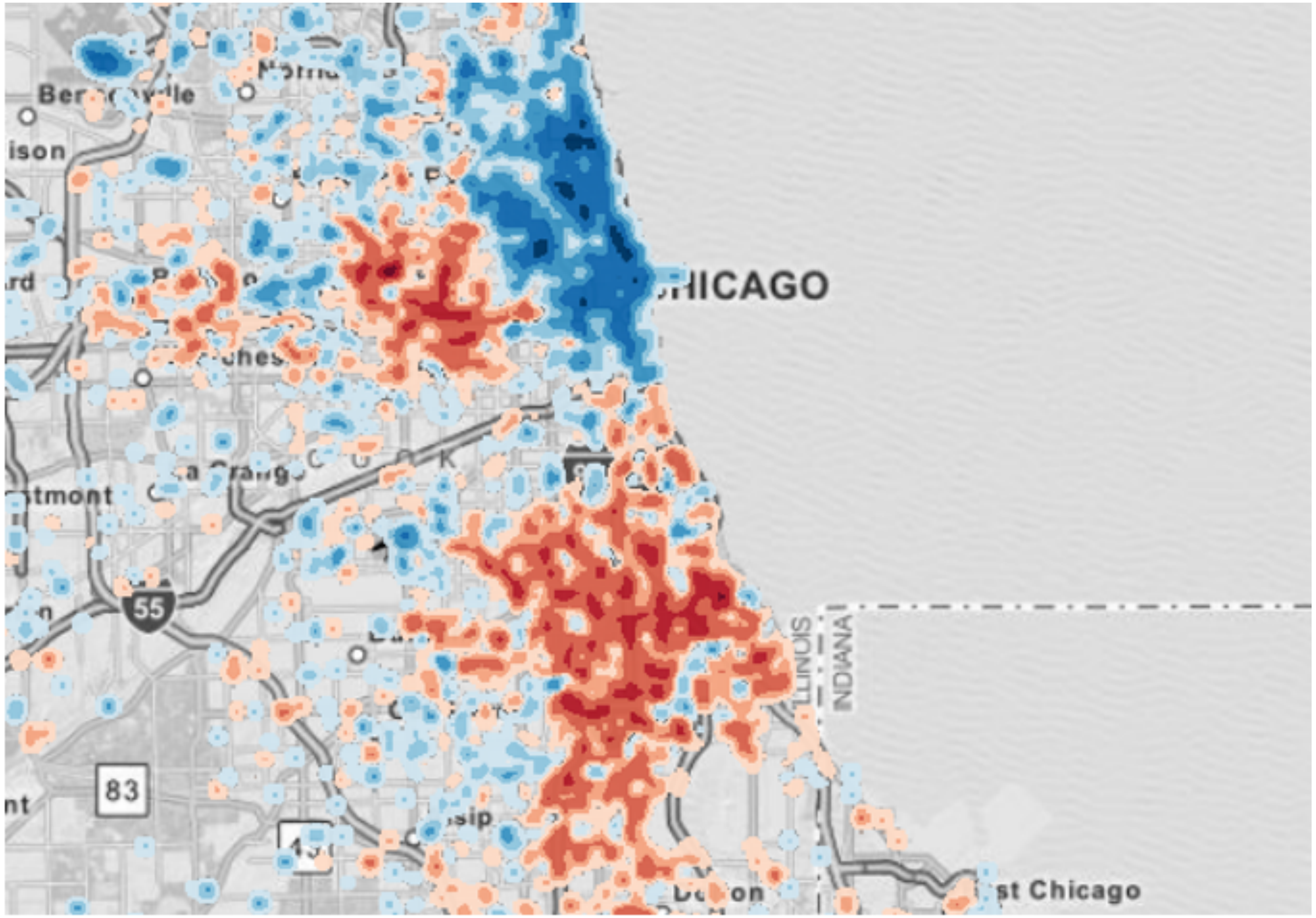
[Lins et. al, 2013]

iPhone vs. Android Map



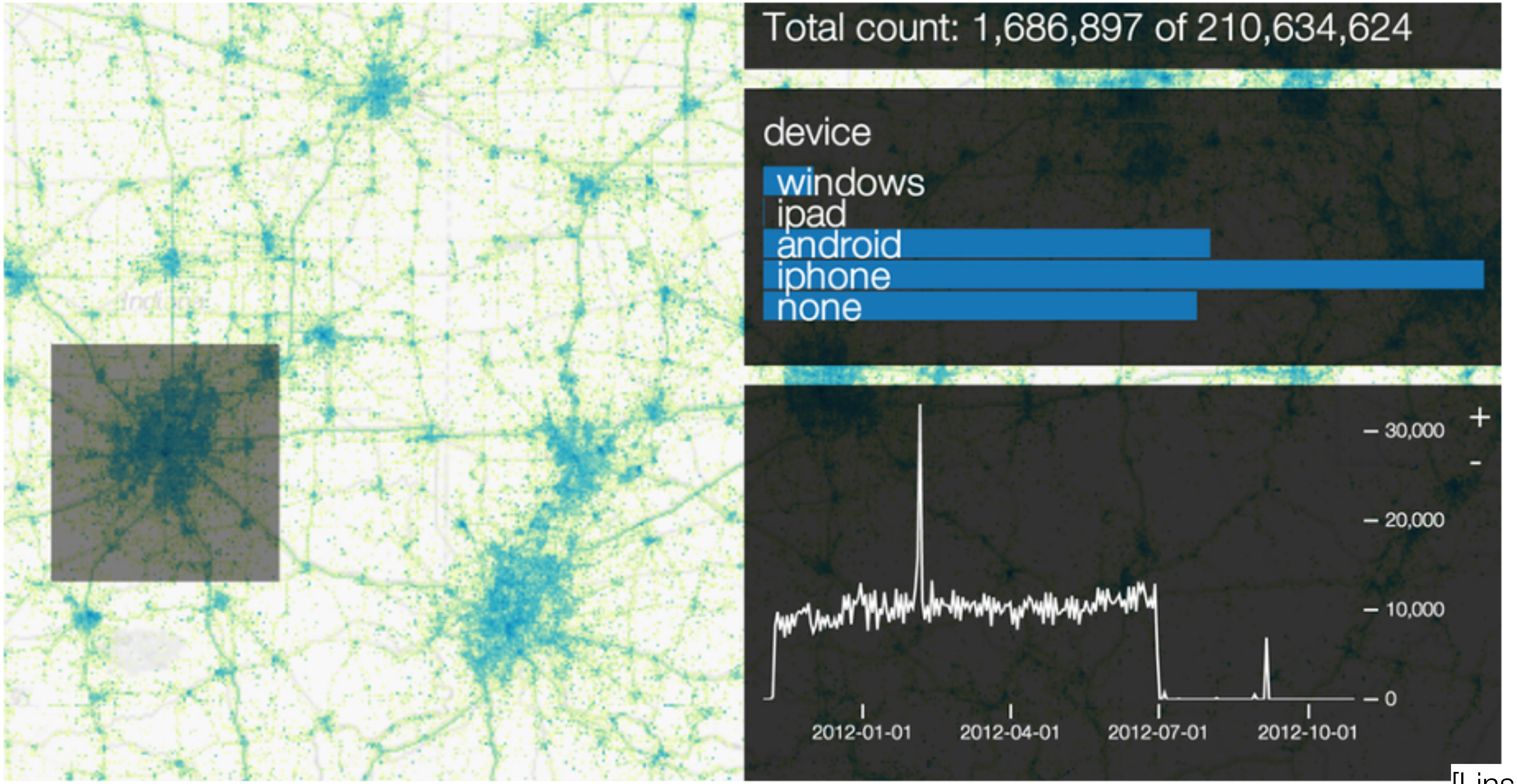
[Lins et. al, 2013]

Zoom into Chicago



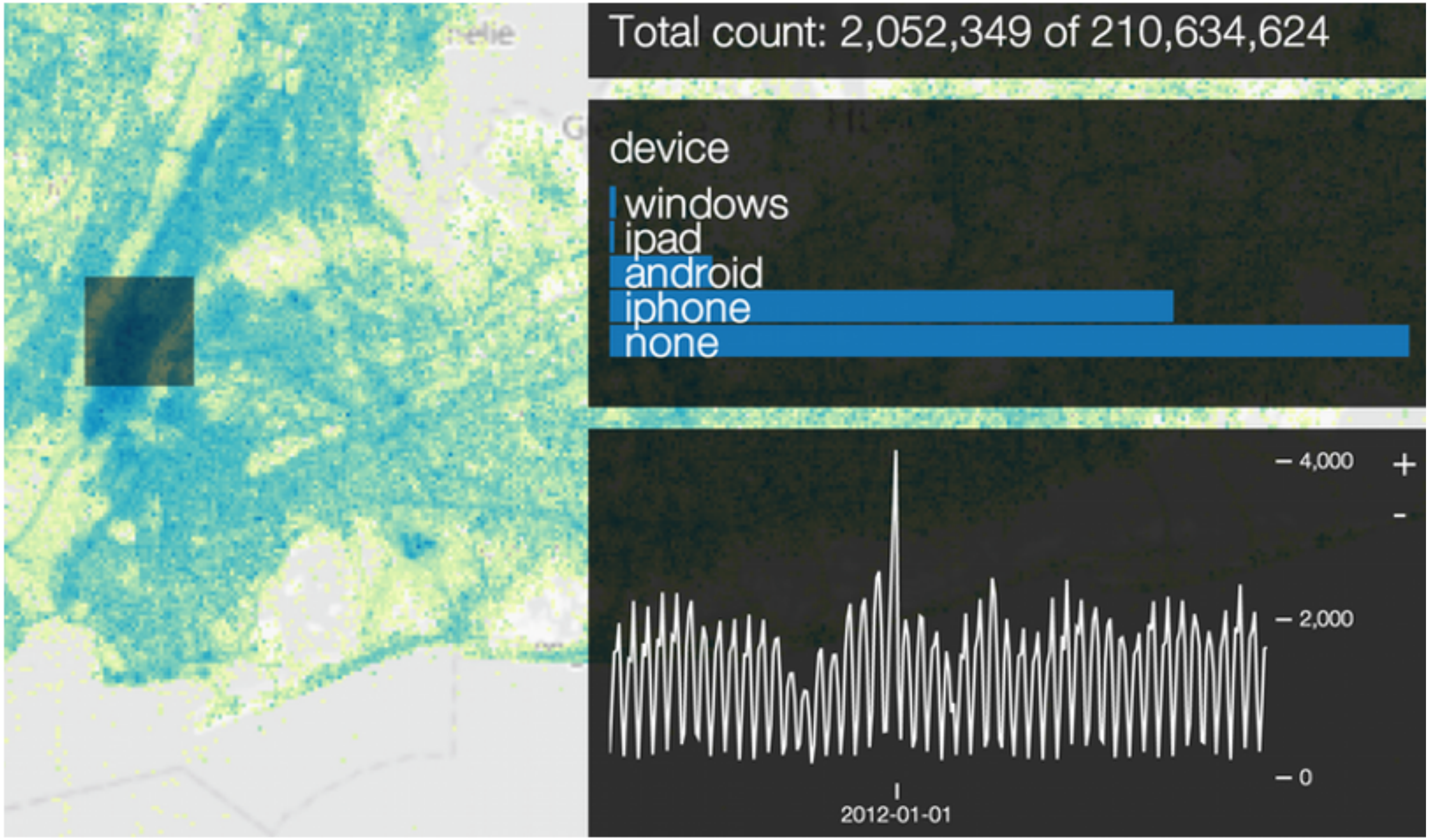
[Lins et. al, 2013]

SuperBowl in Indianapolis



[Lins et. al, 2013]

New Year's Eve in Manhattan

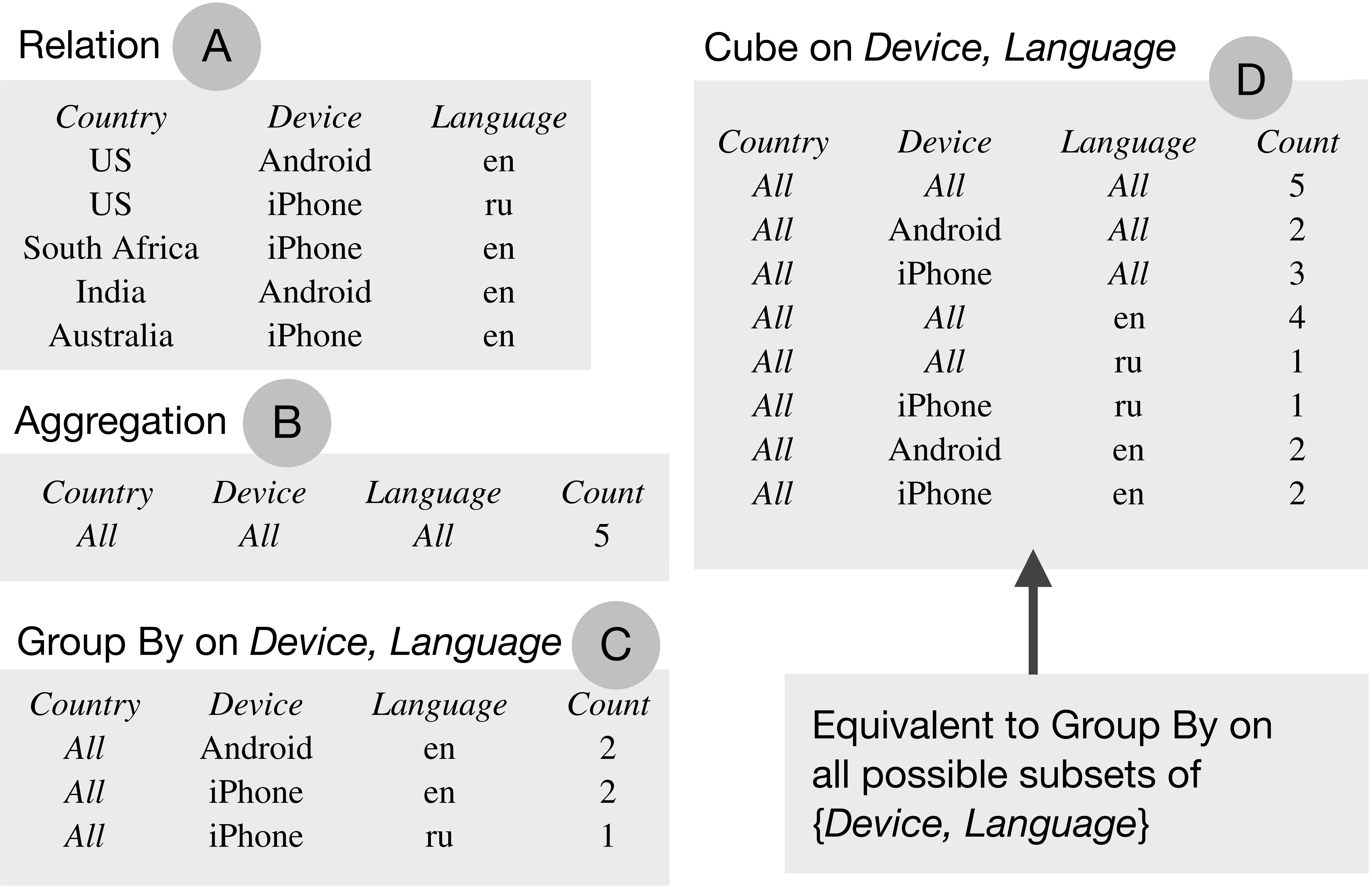


[Lins et. al, 2013]

Aggregations on Spatiotemporal Data

- Spatial: e.g. counting events in a spatial region (world or San Fran.)
- Temporal: e.g. queries at multiple scares (hour, day, week, month)
- Seek to address Visual Information Seeking Mantra:
- Overview first, zoom and filter, details-on-demand
- Multidimensional:
  - Latitude, Longitude, Time + **more**

Data Cube Aggregations



Relation A

Country	*Device*	*Language*
US	Android	en
US	iPhone	ru
South Africa	iPhone	en
India	Android	en
Australia	iPhone	en

Aggregation B

Country	*Device*	*Language*	*Count*
All	*All*	*All*	5

Group By on *Device, Language* C

Country	*Device*	*Language*	*Count*
All	Android	en	2
All	iPhone	en	2
All	iPhone	ru	1

Cube on *Device, Language* D

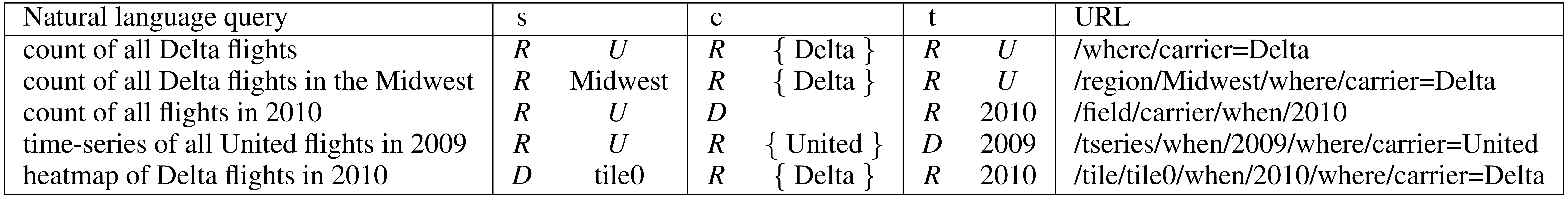
Country	*Device*	*Language*	*Count*
All	*All*	*All*	5
All	Android	*All*	2
All	iPhone	*All*	3
All	*All*	en	4
All	*All*	ru	1
All	iPhone	ru	1
All	Android	en	2
All	iPhone	en	2

Equivalent to Group By on all possible subsets of {*Device, Language*}

[Lins et. al, 2013]

Nanocube Queries

- Representing natural language queries as data cube queries

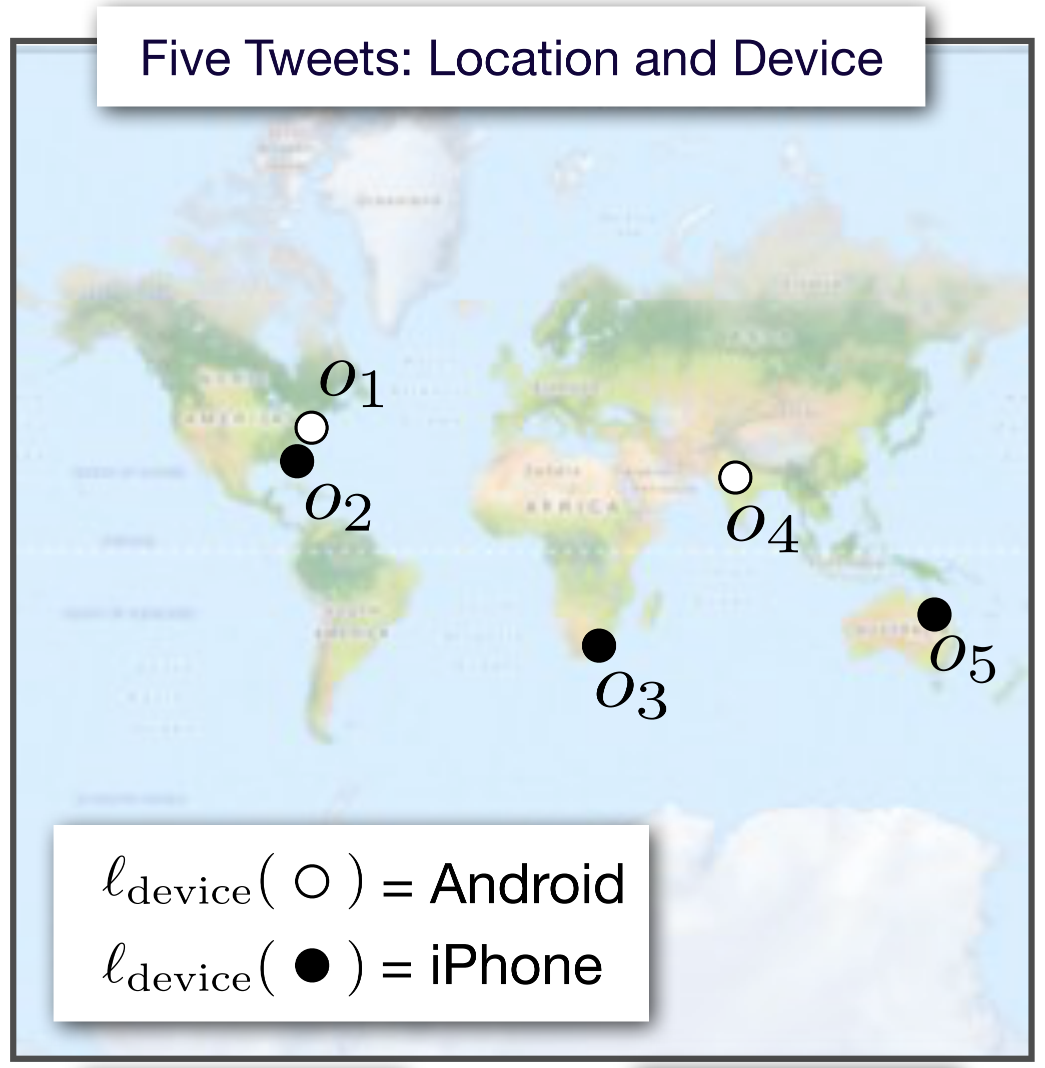


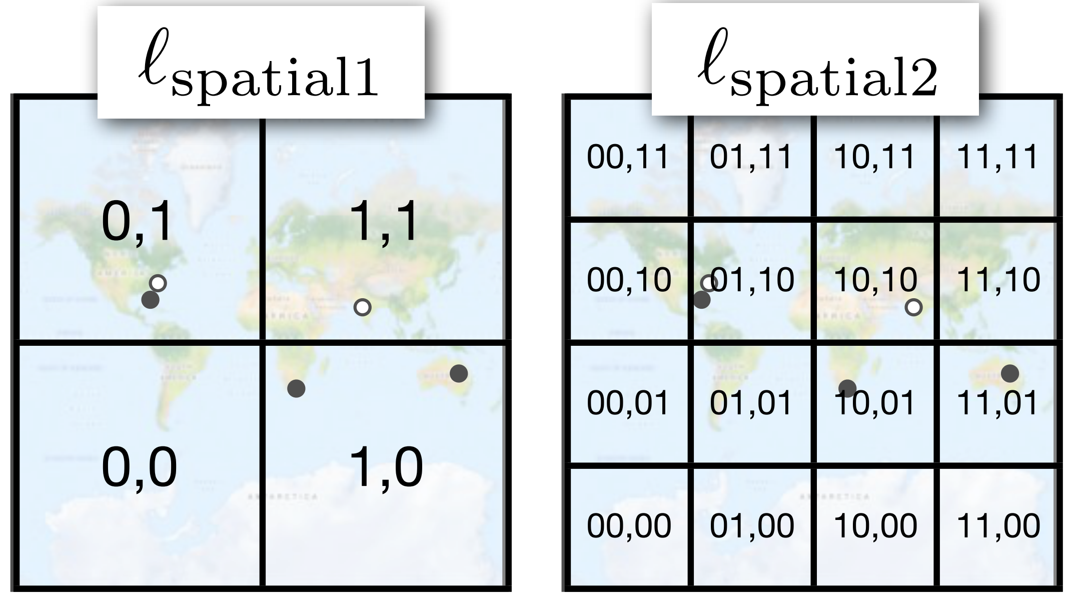
Natural language query	s		c		t		URL
count of all Delta flights	*R*	*U*	*R*	{ Delta }	*R*	*U*	/where/carrier=Delta
count of all Delta flights in the Midwest	*R*	Midwest	*R*	{ Delta }	*R*	*U*	/region/Midwest/where/carrier=Delta
count of all flights in 2010	*R*	*U*	*D*		*R*	2010	/field/carrier/when/2010
time-series of all United flights in 2009	*R*	*U*	*R*	{ United }	*D*	2009	/tseries/when/2009/where/carrier=United
heatmap of Delta flights in 2010	*D*	tile0	*R*	{ Delta }	*R*	2010	/tile/tile0/when/2010/where/carrier=Delta

- s = space, c = category, t = time
- R = rollup, D = drill down
- <value> after RD = subset of dimension's domain, U = universe
- Note that time queries are stored in an array of cumulative counts

[Lins et. al, 2013]

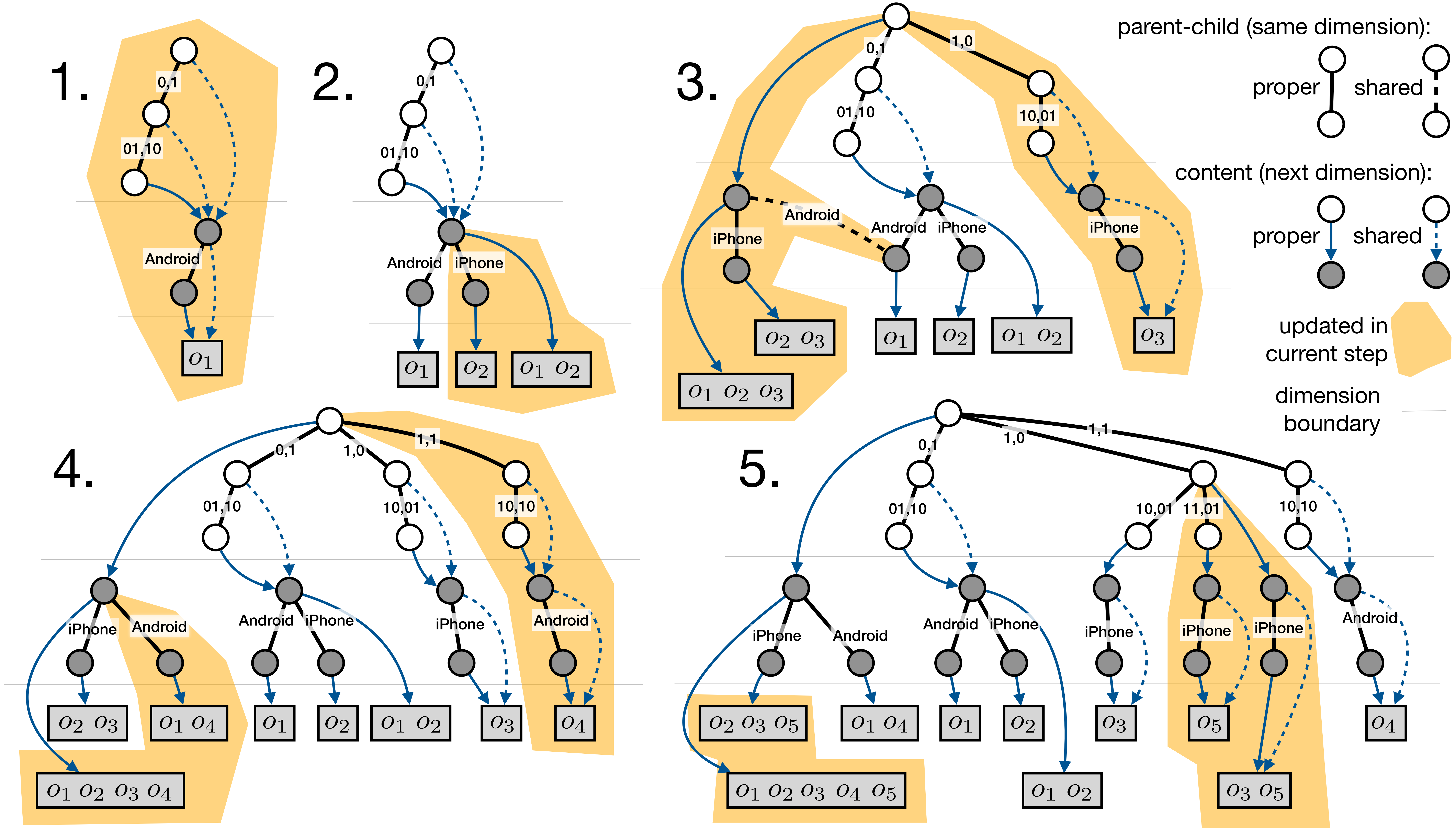
Building a Nanocube


Five Tweets: Location and Device
o_1
o_2
o_3
o_4
o_5
$\ell_{\text{device}}(○)$ = Android
$\ell_{\text{device}}(●)$ = iPhone



Indexing Schema

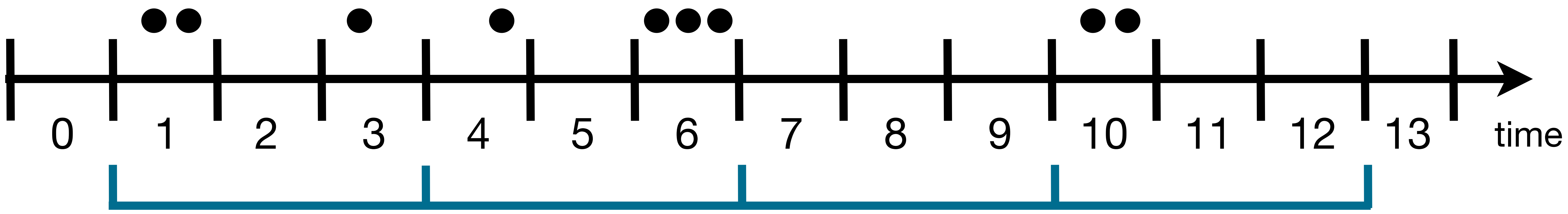
$$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$$



[Lins et. al, 2013]

Summed-area Table

- Every node in the previous figure stores an array of timestamped counts like this:



query/tseries/1/3/4

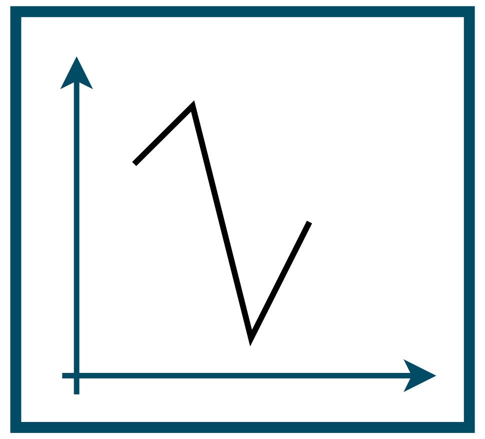
start at bin 1, use buckets of 3 bins each, and collect 4 of these buckets

solve using...

bin: 1 accum: 2	bin: 3 accum: 3	bin: 4 accum: 4	bin: 6 accum: 7	bin: 10 accum: 9

A Summed Table Sparse Representation for Counts

result

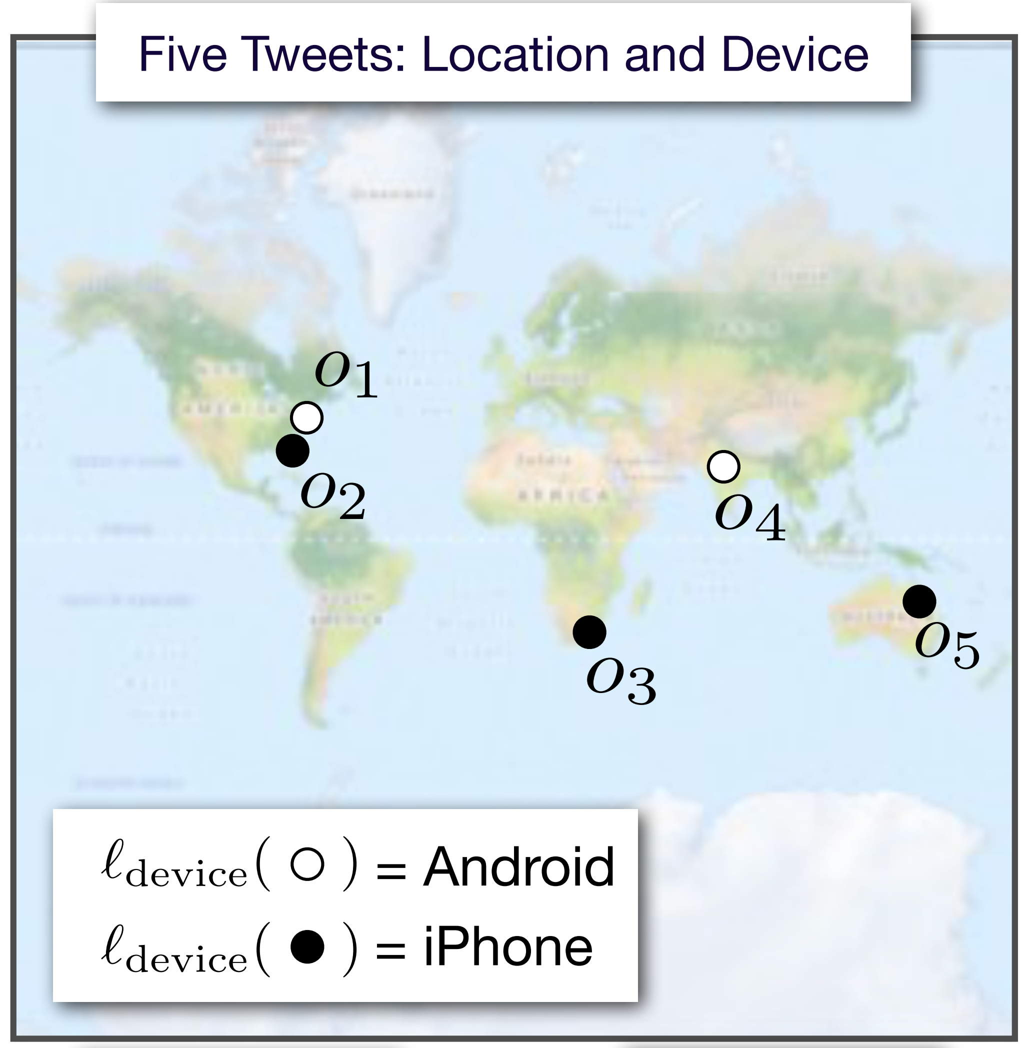


3	4	0	2

[Lins et. al, 2013]

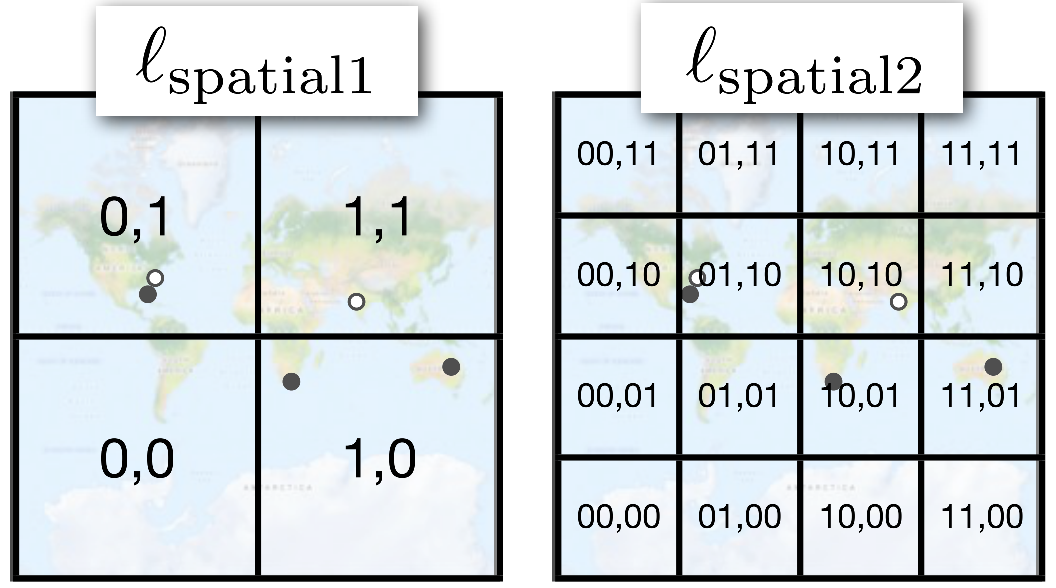
Building a Nanocube: Step 1

Five Tweets: Location and Device



ℓ_{device}(○) = Android

ℓ_{device}(●) = iPhone



ℓ_{spatial1}

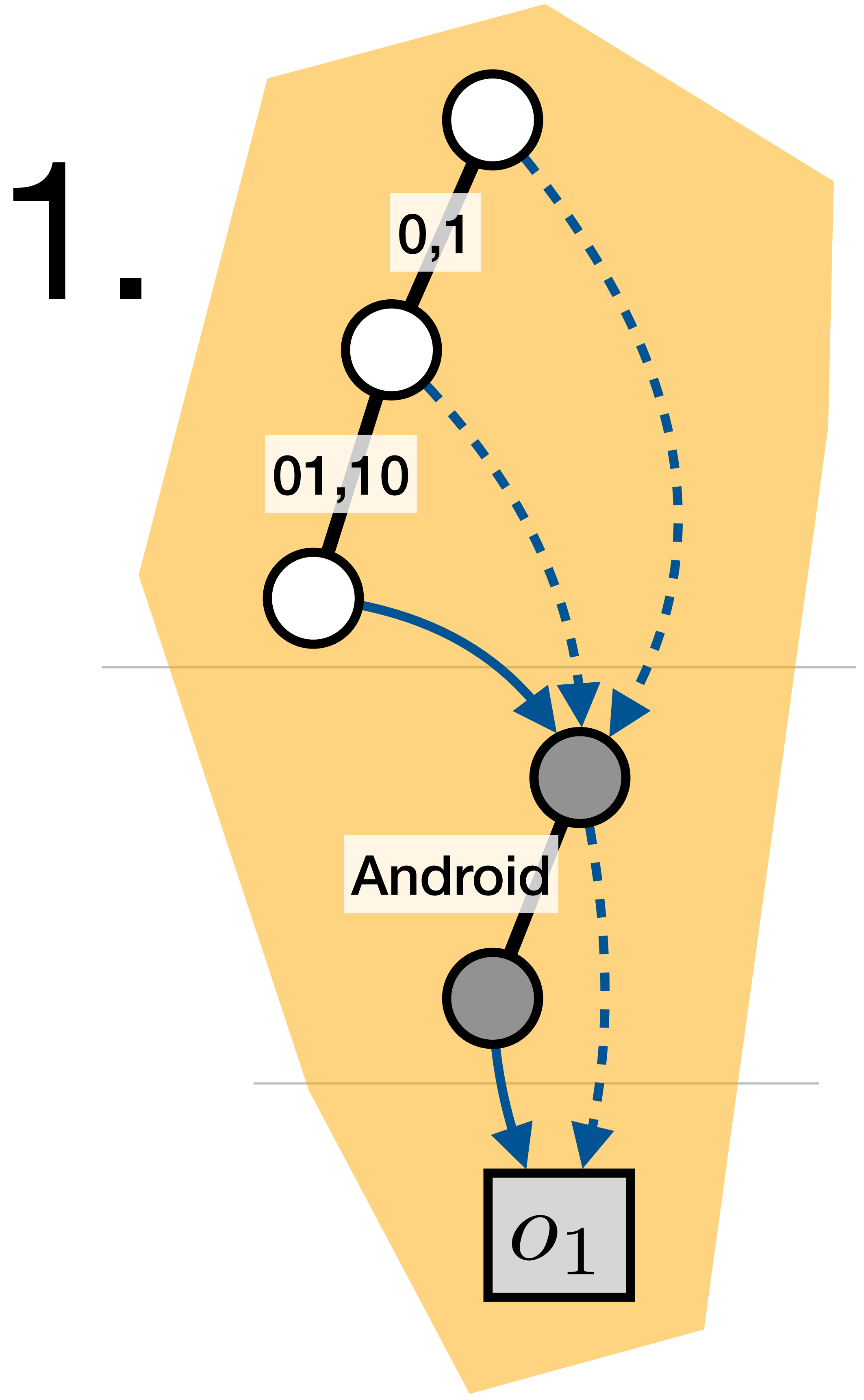
0,1	1,1
0,0	1,0

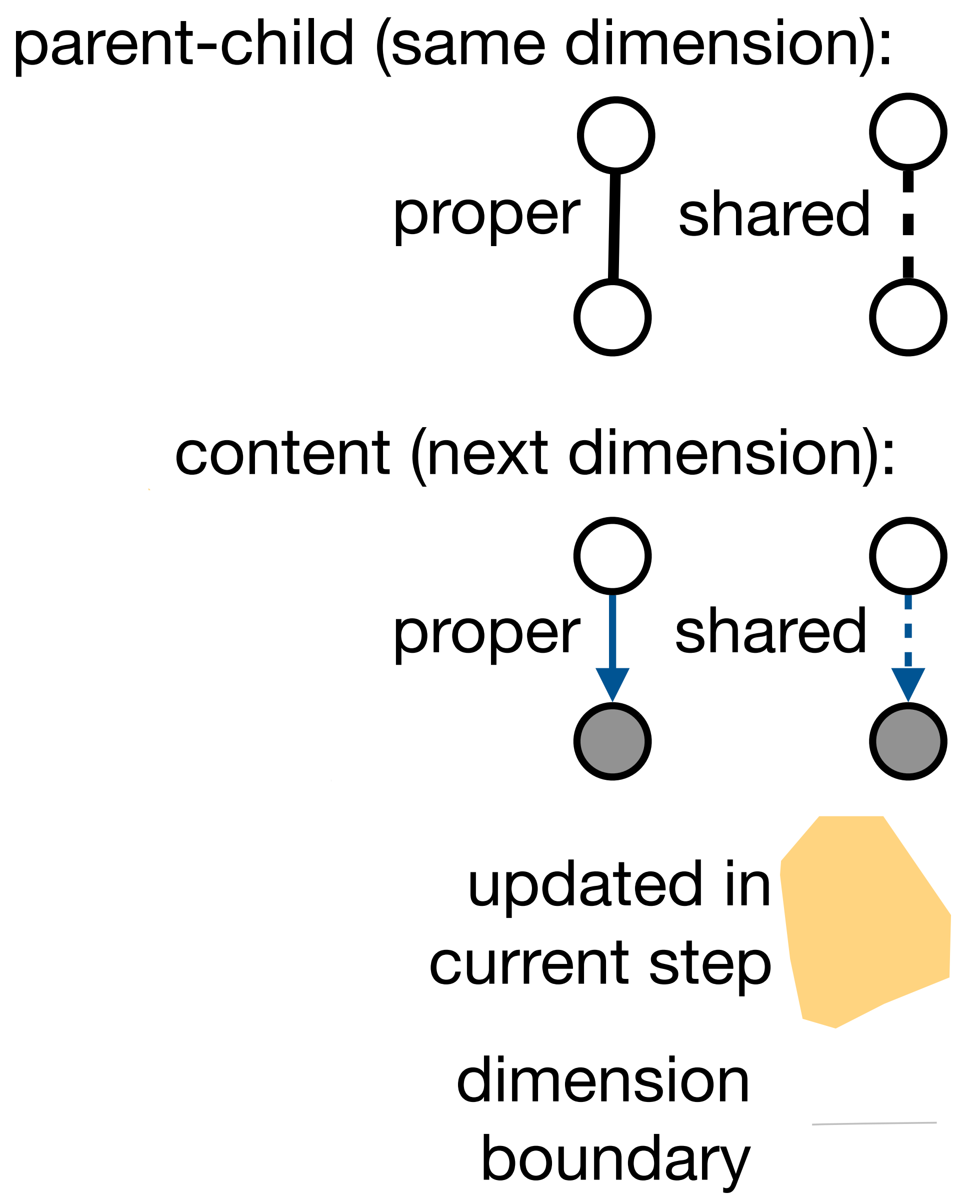
ℓ_{spatial2}

00,11	01,11	10,11	11,11
00,10	01,10	10,10	11,10
00,01	01,01	10,01	11,01
00,00	01,00	10,00	11,00

Indexing Schema

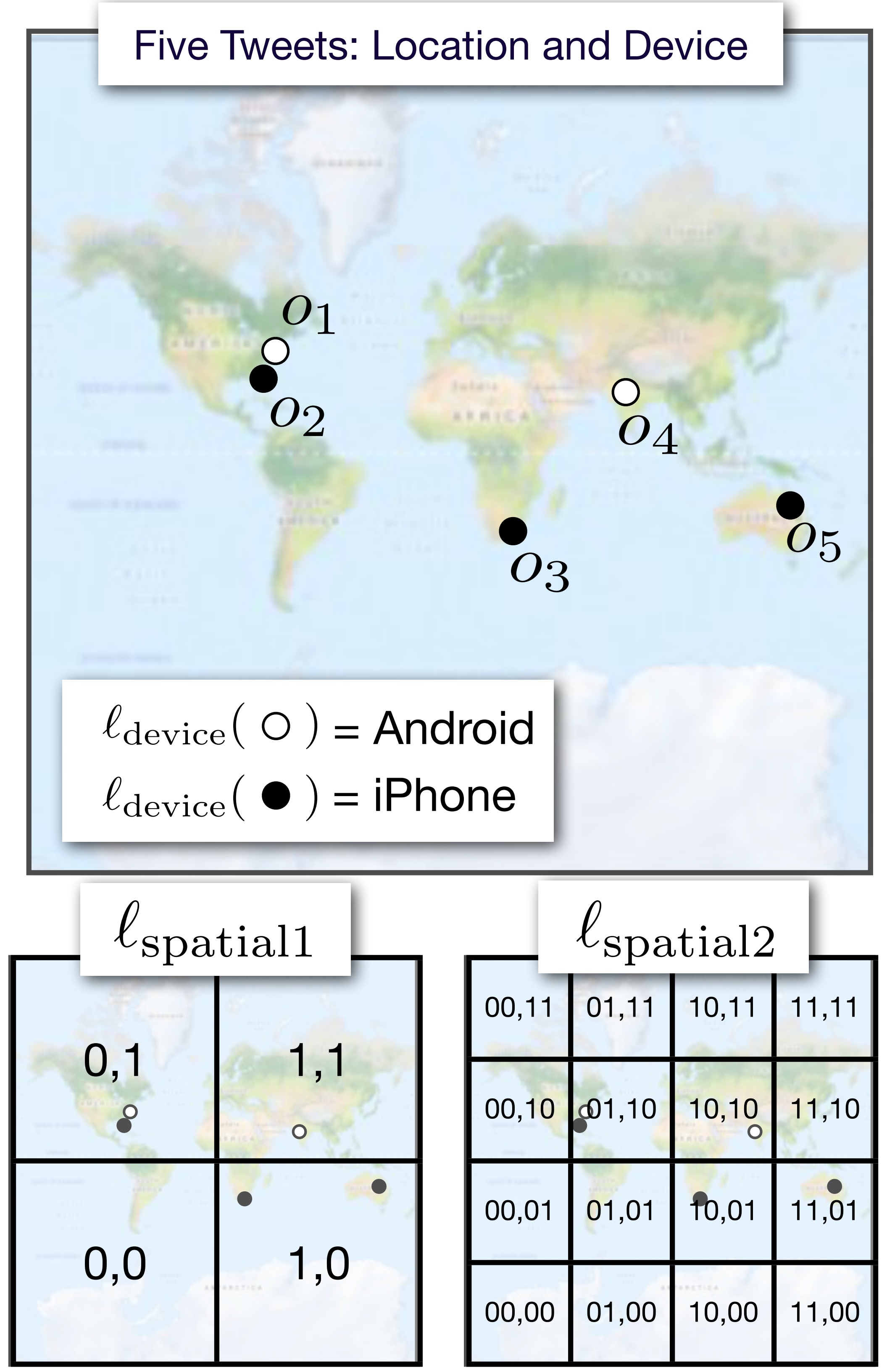
$$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$$





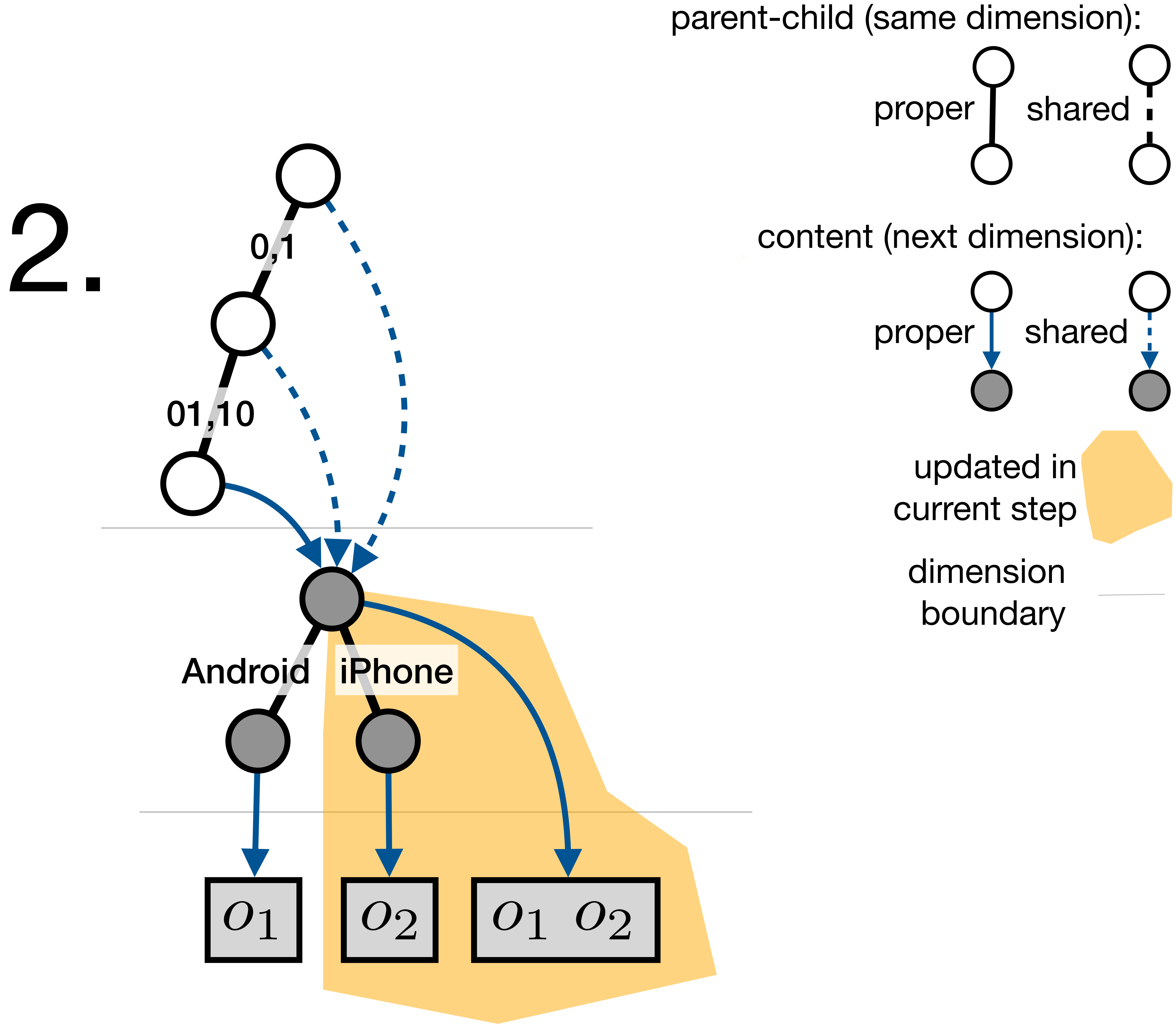
[Lins et. al, 2013]

Building a Nanocube: Step 2



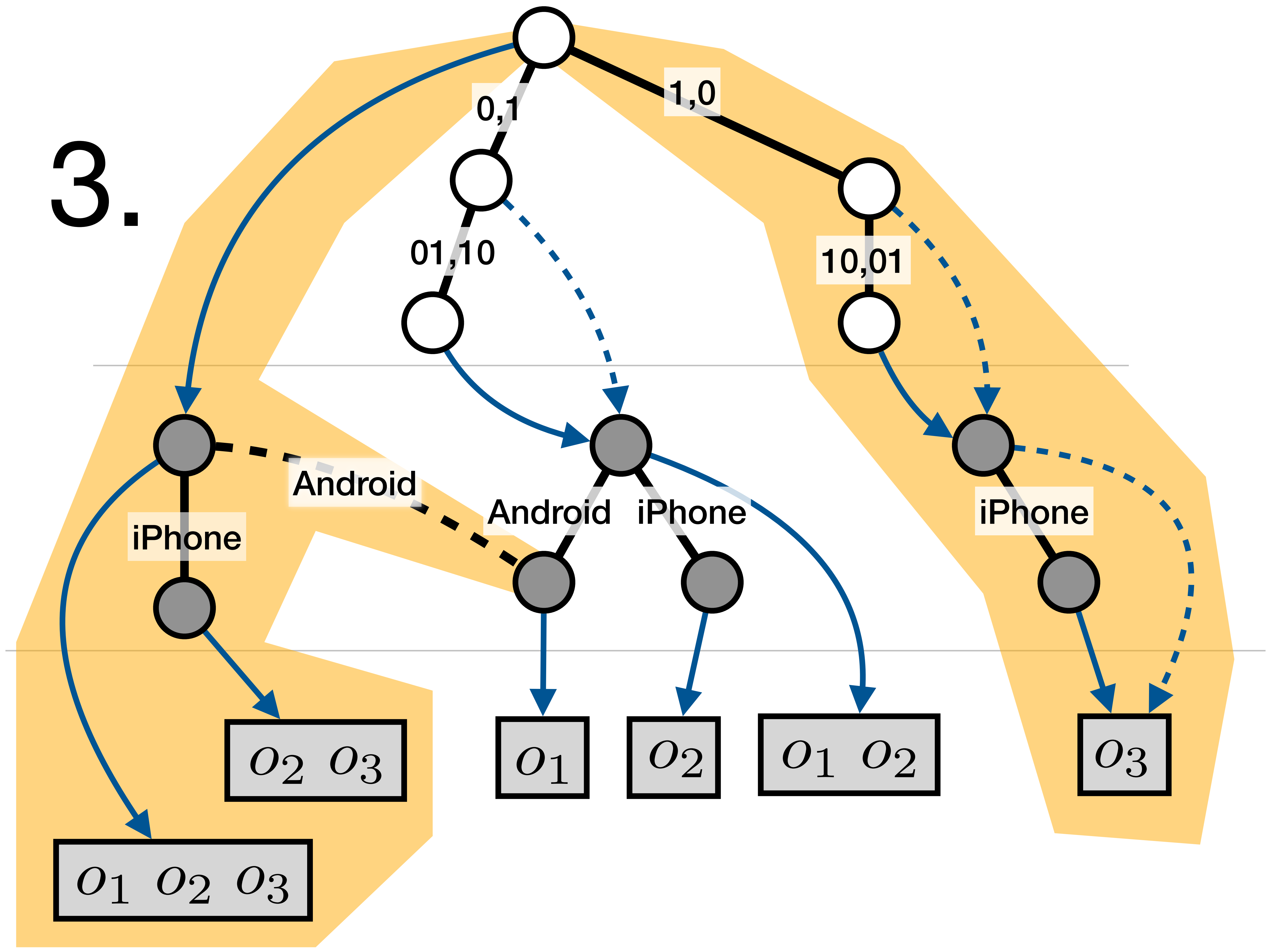
Indexing Schema

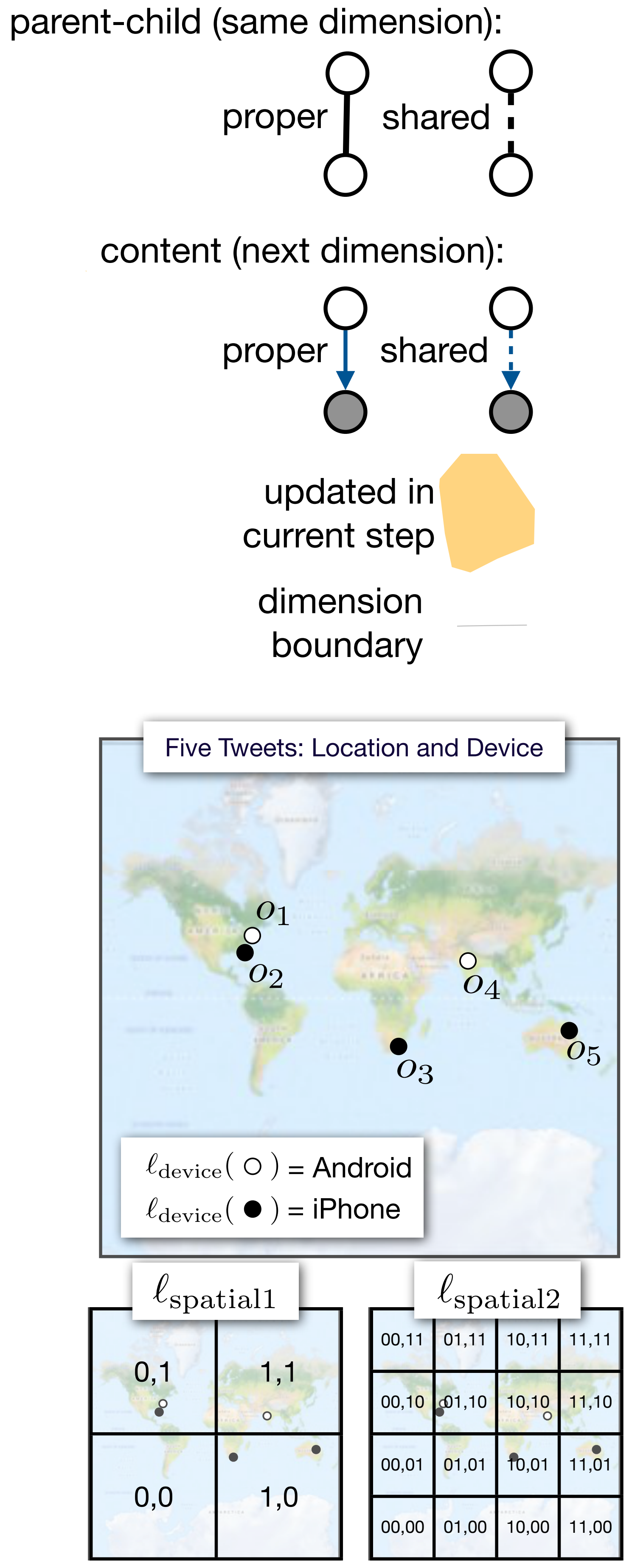
$$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$$



[Lins et. al, 2013]

Building a Nanocube: Step 3



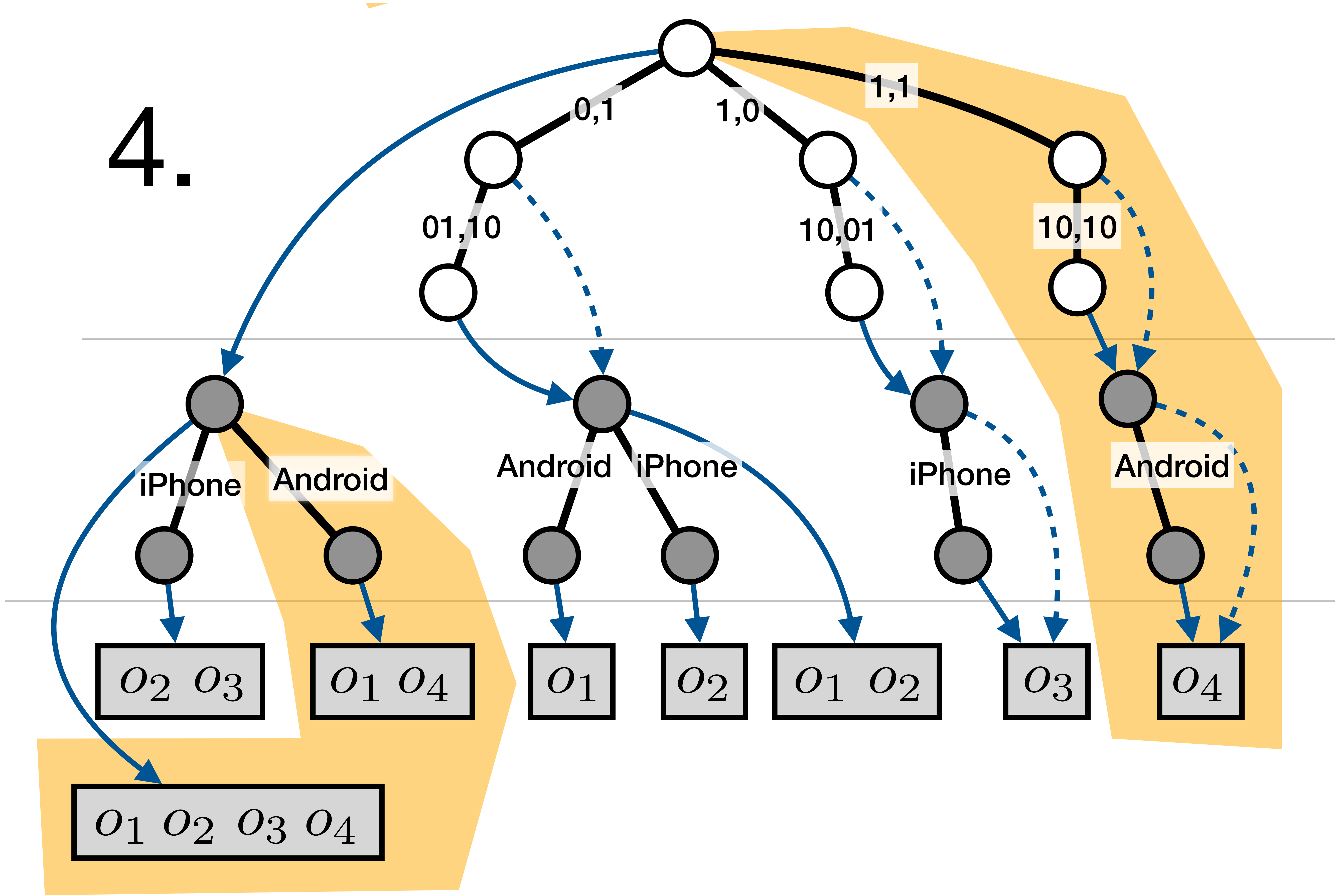


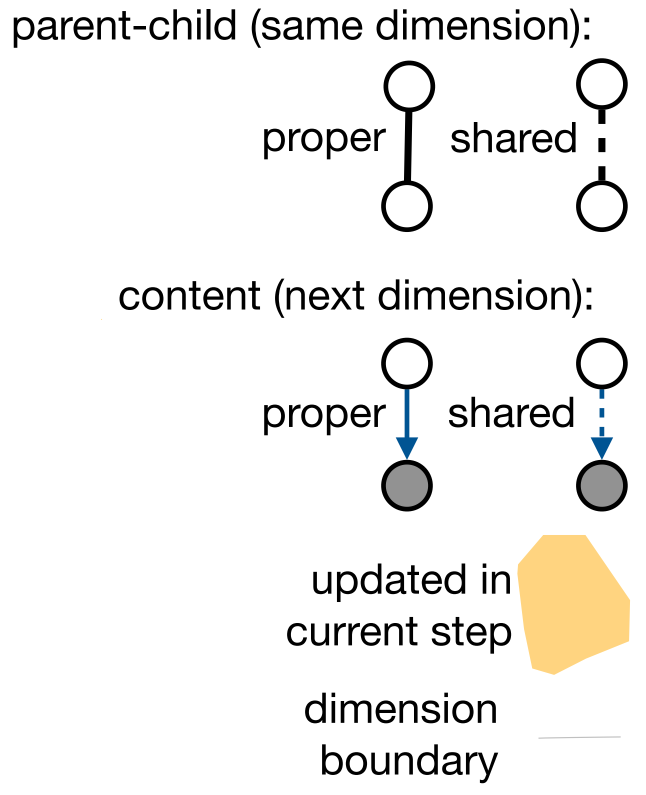
Indexing Schema

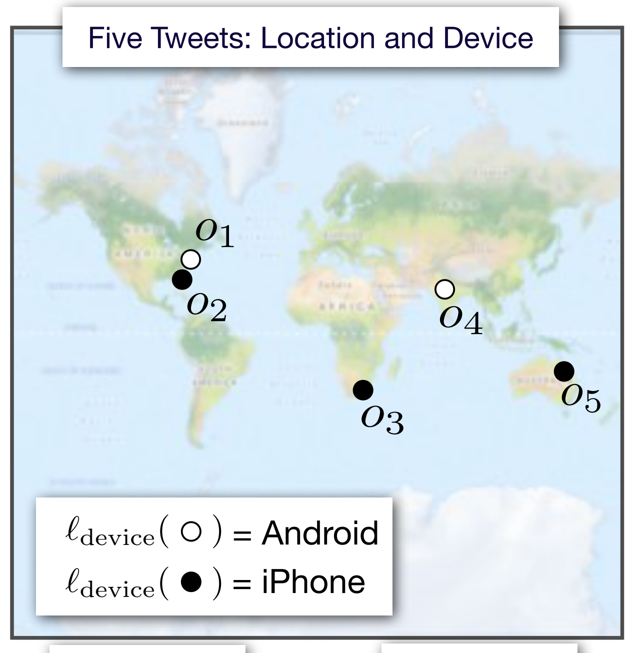
$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$

[Lins et. al, 2013]

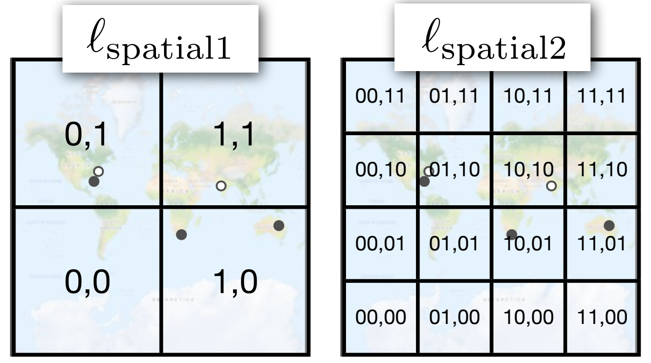
Building a Nanocube: Step 4







Five Tweets: Location and Device
o_1
o_2
o_3
o_4
o_5
$\ell_{\text{device}}(\ \bigcirc\ )$ = Android
$\ell_{\text{device}}(\ \bullet\ )$ = iPhone

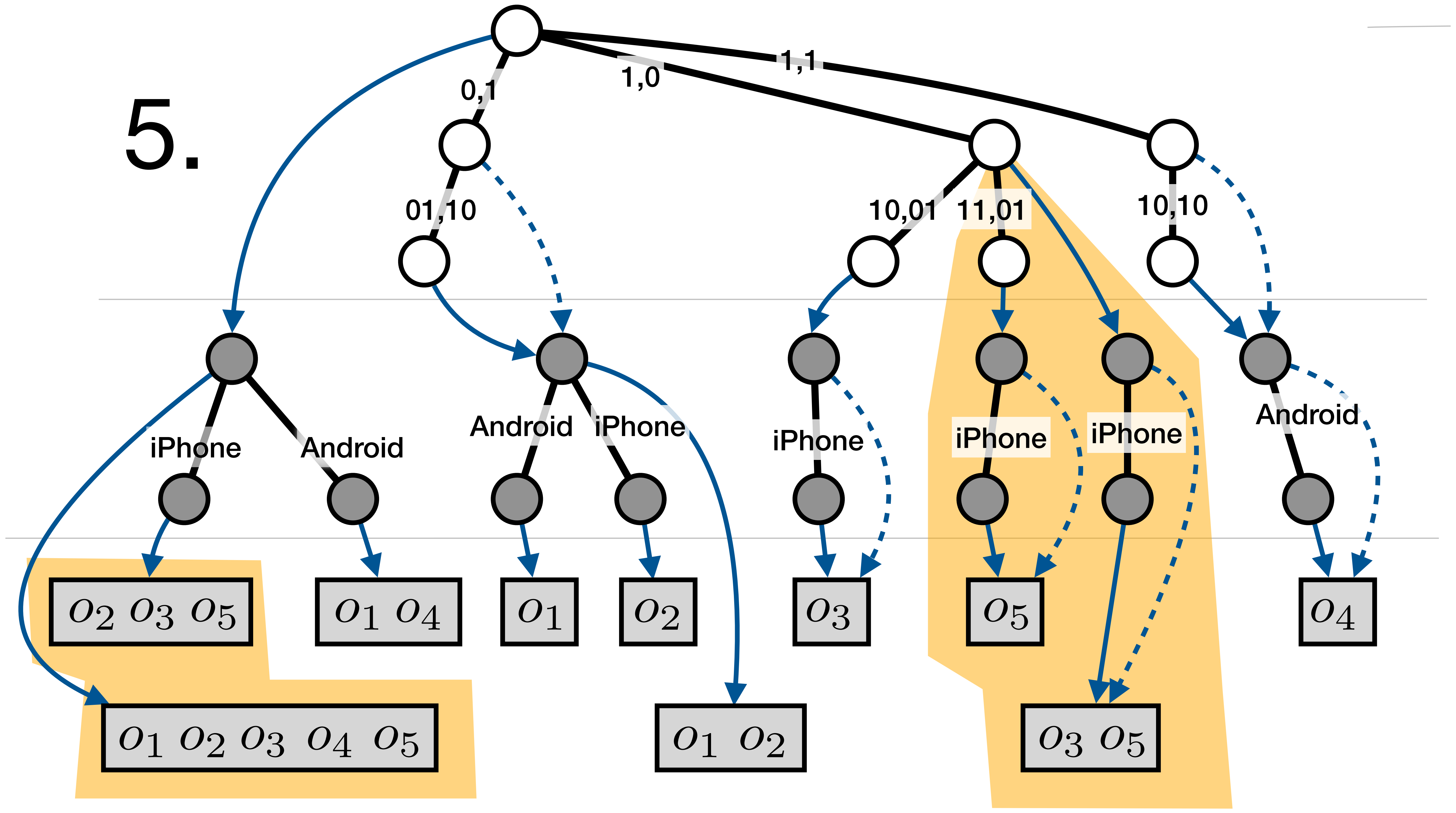


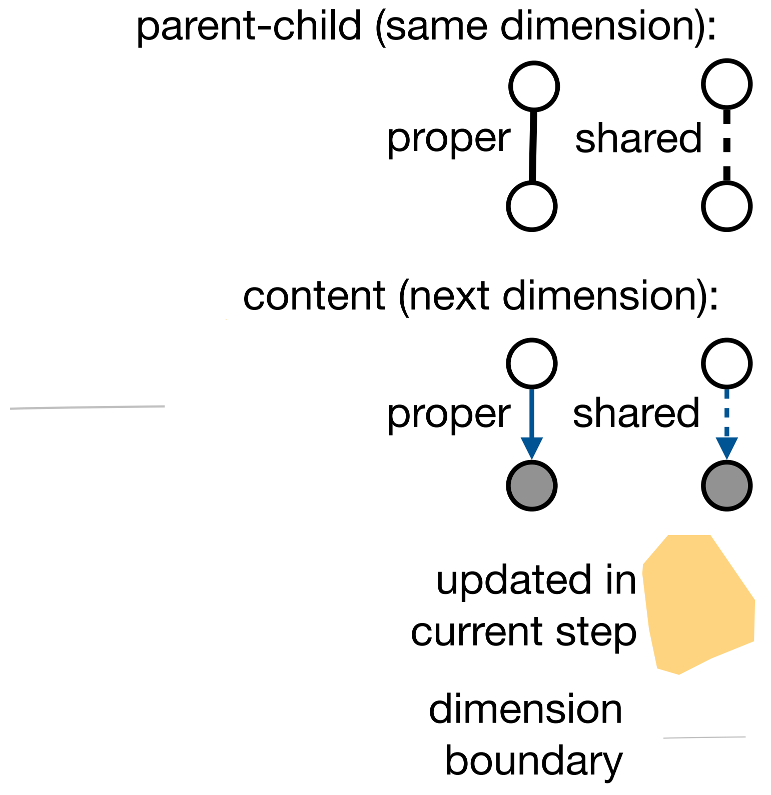
Indexing Schema

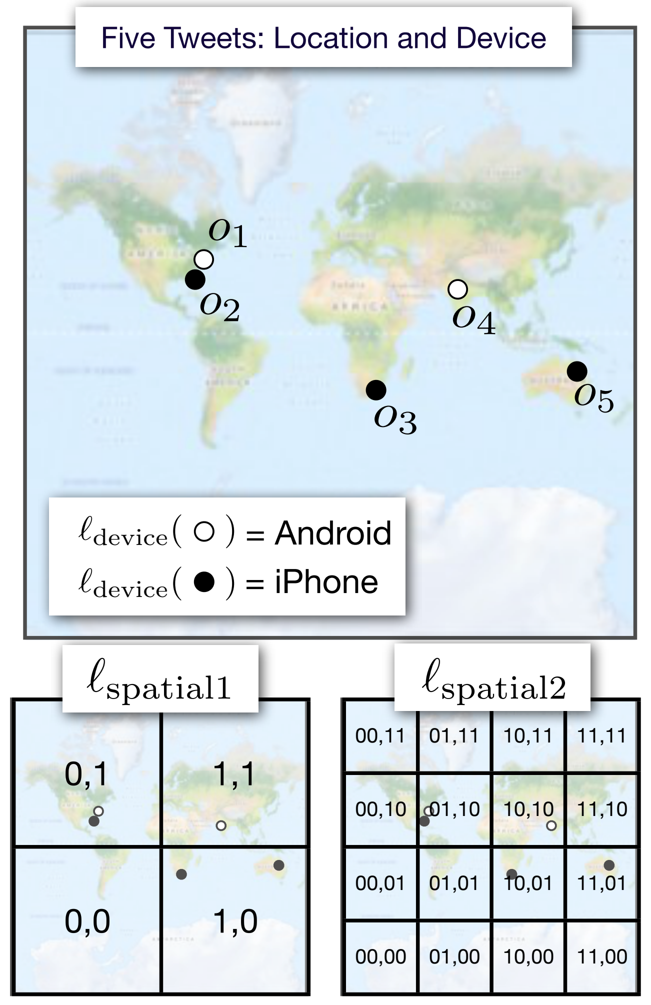
$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$

[Lins et. al, 2013]

Building a Nanocube: Step 5






Five Tweets: Location and Device
o_1
o_2
o_3
o_4
o_5
$\ell_{\text{device}}(\ ○\ )$ = Android
$\ell_{\text{device}}(\ ●\ )$ = iPhone
ℓ_{spatial1}
ℓ_{spatial2}

ℓ_{spatial1}

0,1	1,1
0,0	1,0

ℓ_{spatial2}

00,11	01,11	10,11	11,11
00,10	01,10	10,10	11,10
00,01	01,01	10,01	11,01
00,00	01,00	10,00	11,00

Indexing Schema

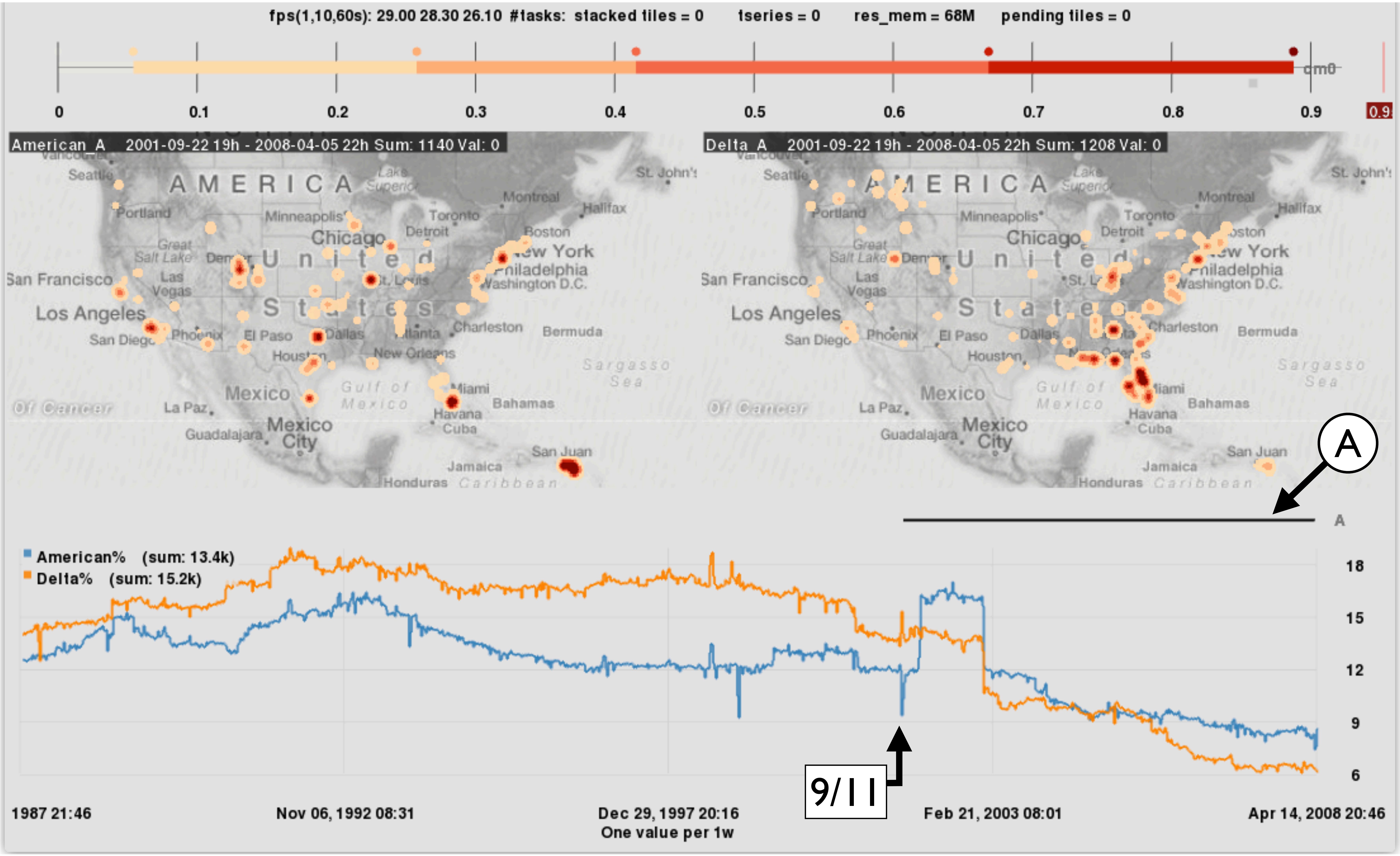
$S = [\,[\ell_{\text{spatial1}}, \ell_{\text{spatial2}}], [\ell_{\text{device}}]\,]$

[Lins et. al, 2013]

Nanocubes Discussion

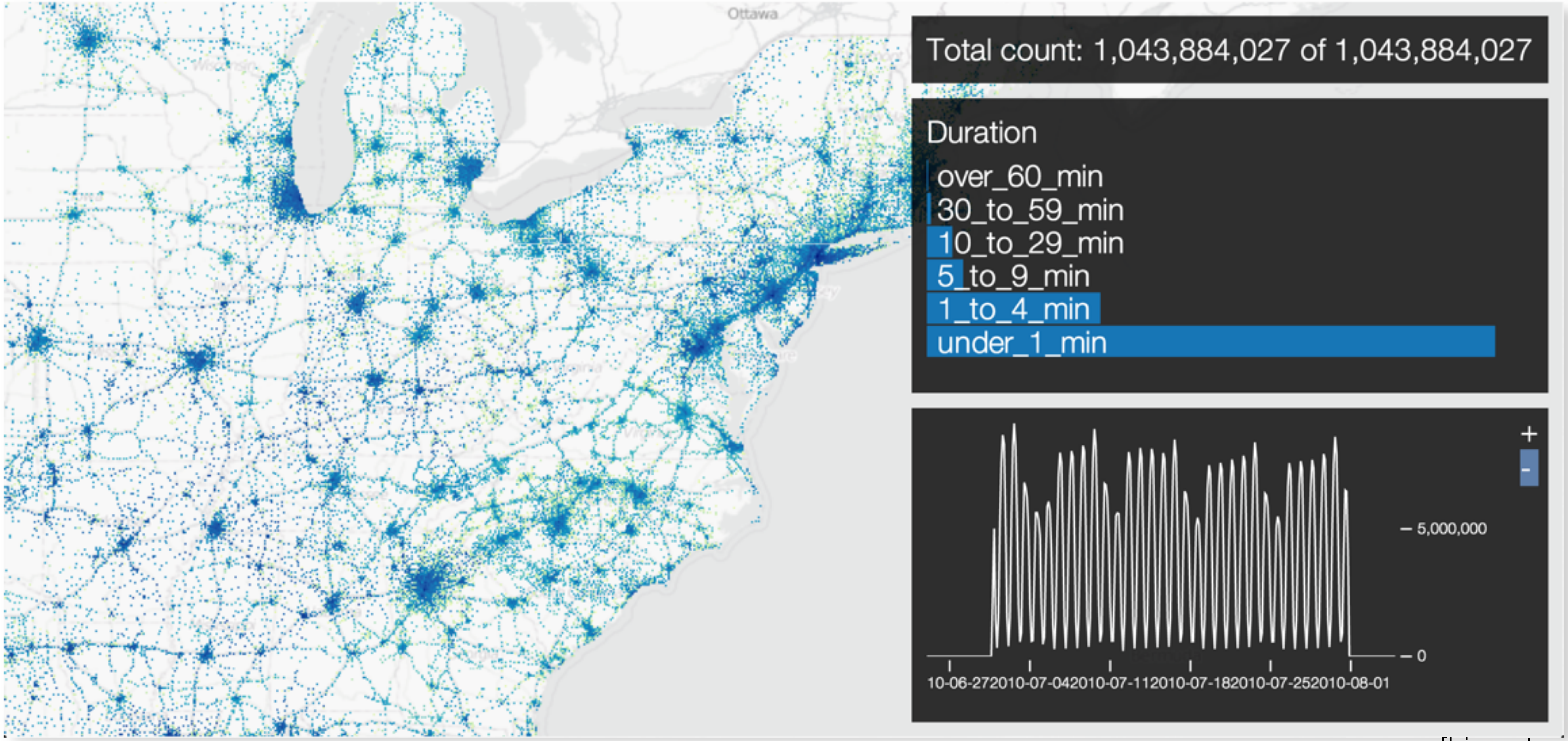
- Save space by organizing the data in a manner that takes advantage of data sparseness
- Limited to one spatial dimension, one temporal dimension
- Precompute **once**, then exploration has **low latency**

Example: American vs. Delta


fps(1,10,60s): 29.00 28.30 26.10 #tasks: stacked tiles = 0 tseries = 0 res_mem = 68M pending tiles = 0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0.9
American_A 2001-09-22 19h - 2008-04-05 22h Sum: 1140 Val: 0
Delta_A 2001-09-22 19h - 2008-04-05 22h Sum: 1208 Val: 0
A
American% (sum: 13.4k)
Delta% (sum: 15.2k)
18
15
12
9
6
1987 21:46
Nov 06, 1992 08:31
Dec 29, 1997 20:16
One value per 1w
Feb 21, 2003 08:01
Apr 14, 2008 20:46
9/11

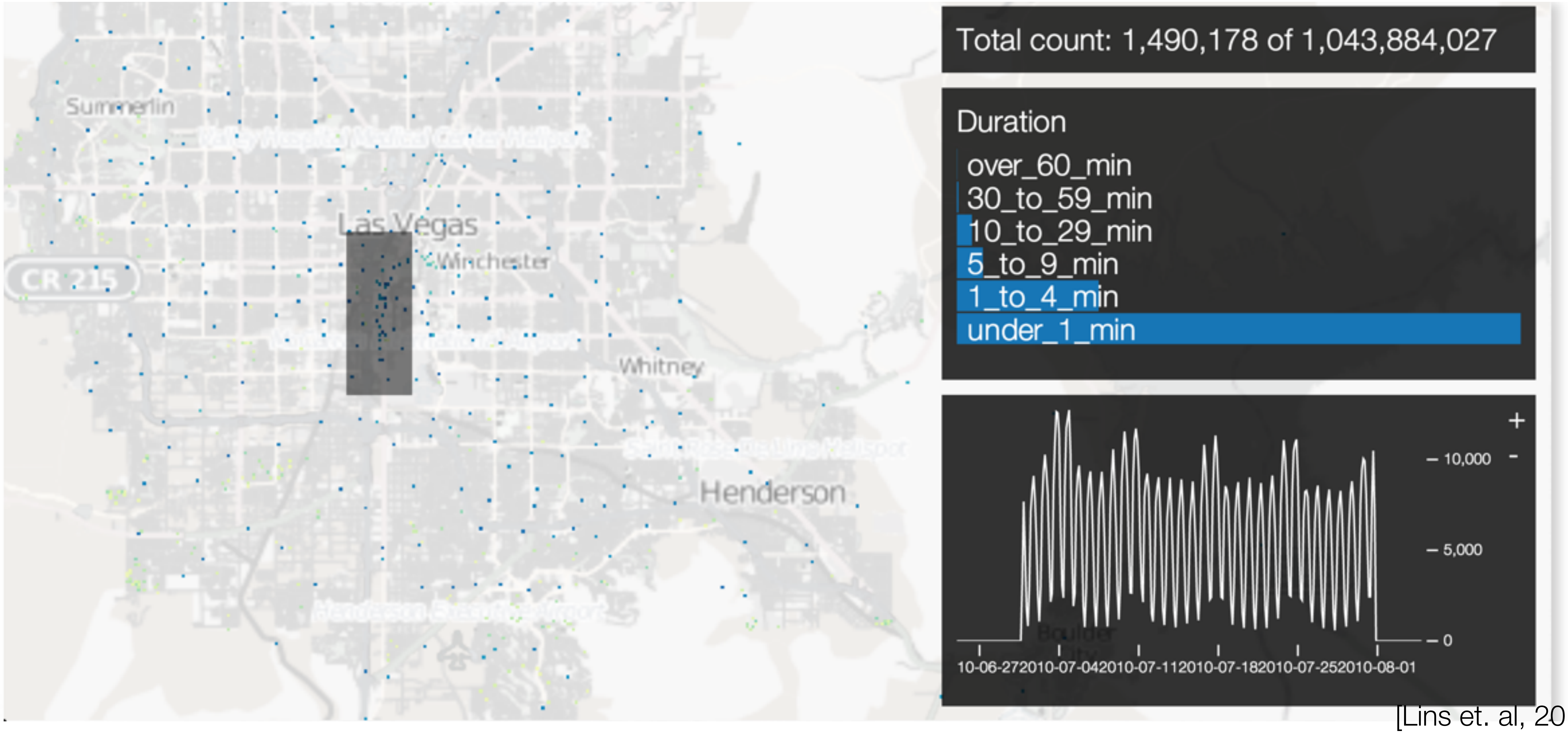
[Lins et. al, 2013]

Example: Cell Data Records



[Lins et. al, 2013]

Example: Cell Data Records



Total count: 1,490,178 of 1,043,884,027

Duration

over_60_min
30_to_59_min
10_to_29_min
5_to_9_min
1_to_4_min
under_1_min

[Lins et. al, 2013]

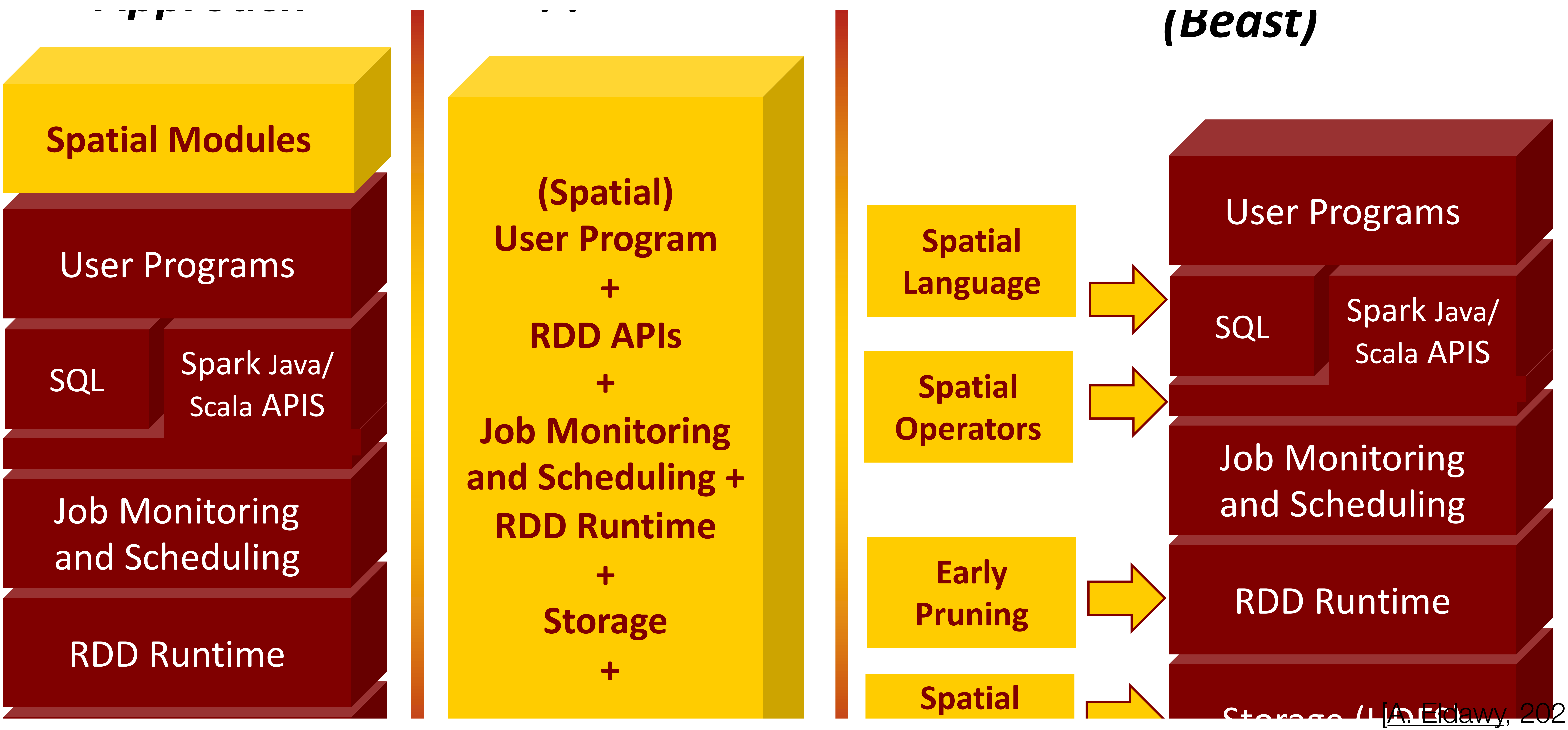
Big Spatial Data Management

A. Eldawy

Motivation

- Complexity of geospatial file formats and problems with reading in a distributed context
- Load balancing the processing across nodes
- Existing solutions focus on one dataset instead of many
- Need a visual-based interface

Beast Architecture



(Beast)

Spatial Modules

User Programs

SQL

Spark Java/ Scala APIS

Job Monitoring and Scheduling

RDD Runtime

(Spatial) User Program + RDD APIs + Job Monitoring and Scheduling + RDD Runtime + Storage +

Spatial Language

Spatial Operators

Early Pruning

Spatial

User Programs

SQL

Spark Java/ Scala APIS

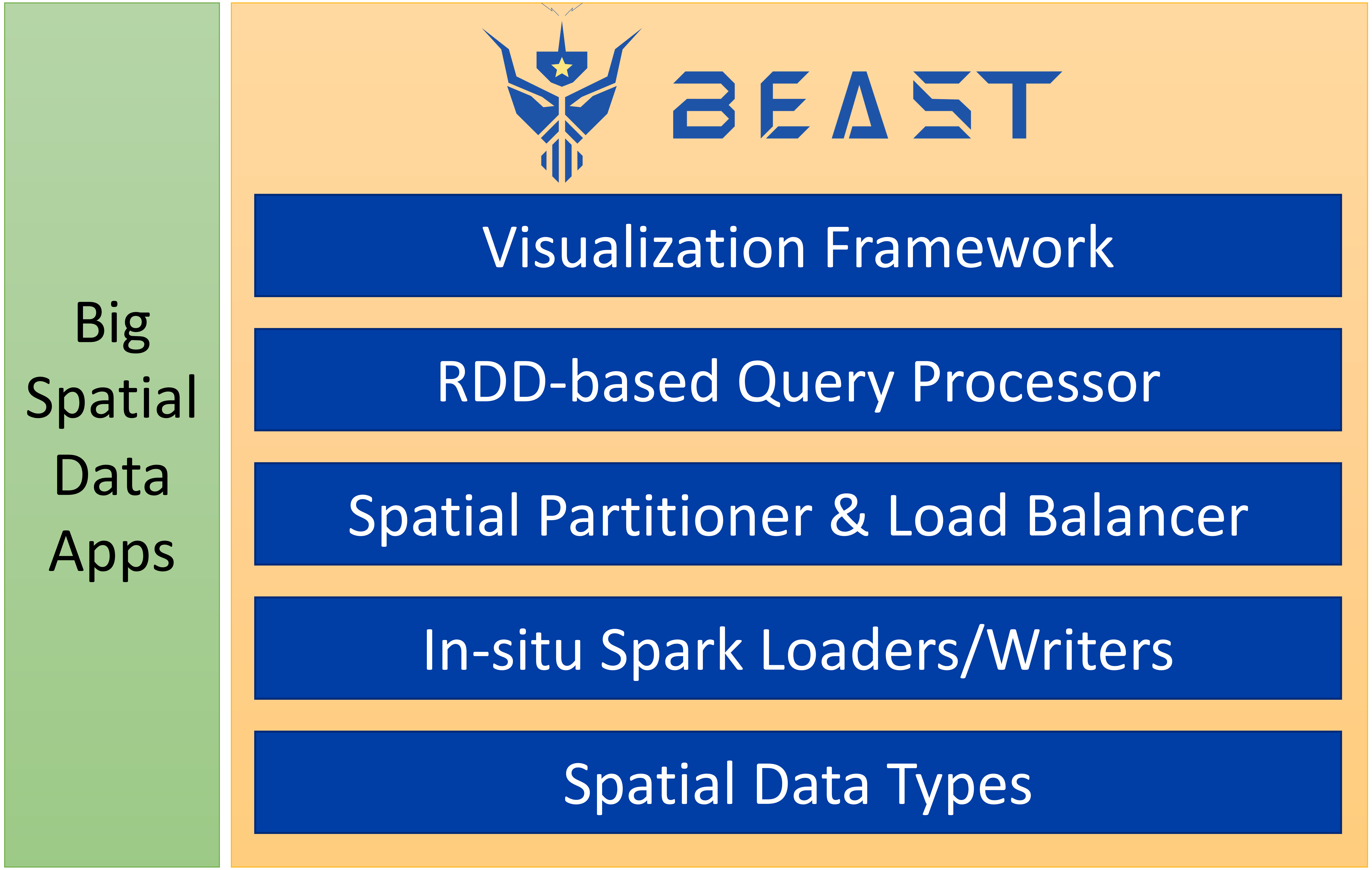
Job Monitoring and Scheduling

RDD Runtime

Storage (H

[A. Eldawy, 2021]

Beast Architecture



Big Spatial Data Apps

BEAST

Visualization Framework

RDD-based Query Processor

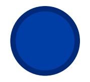
Spatial Partitioner & Load Balancer

In-situ Spark Loaders/Writers

Spatial Data Types

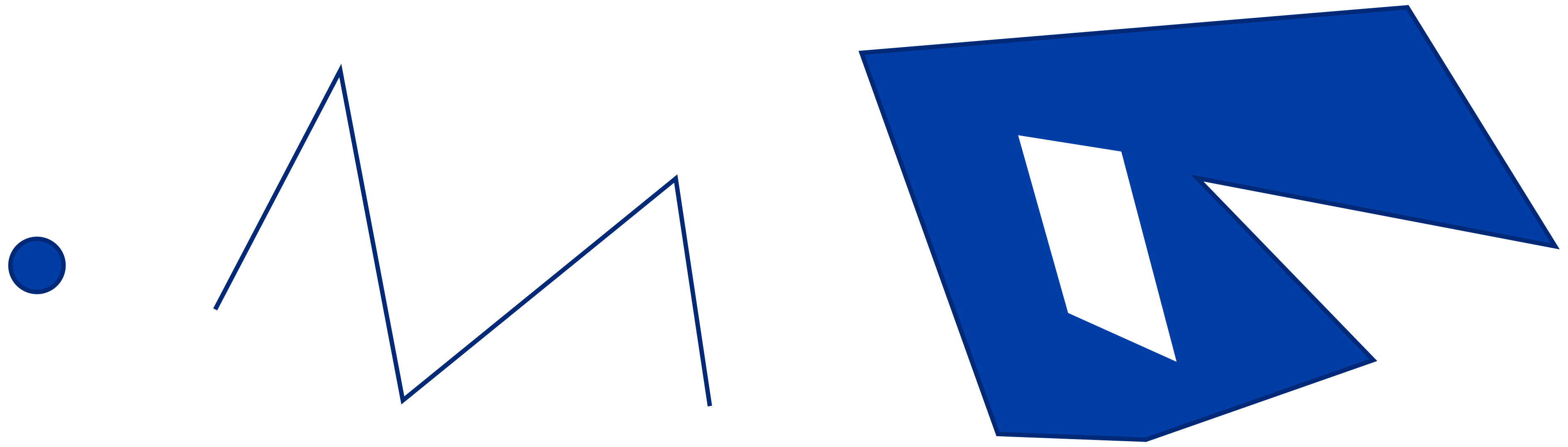
[A. Eldawy, 2021]

Beast Spatial Data Types



Point

Envelope



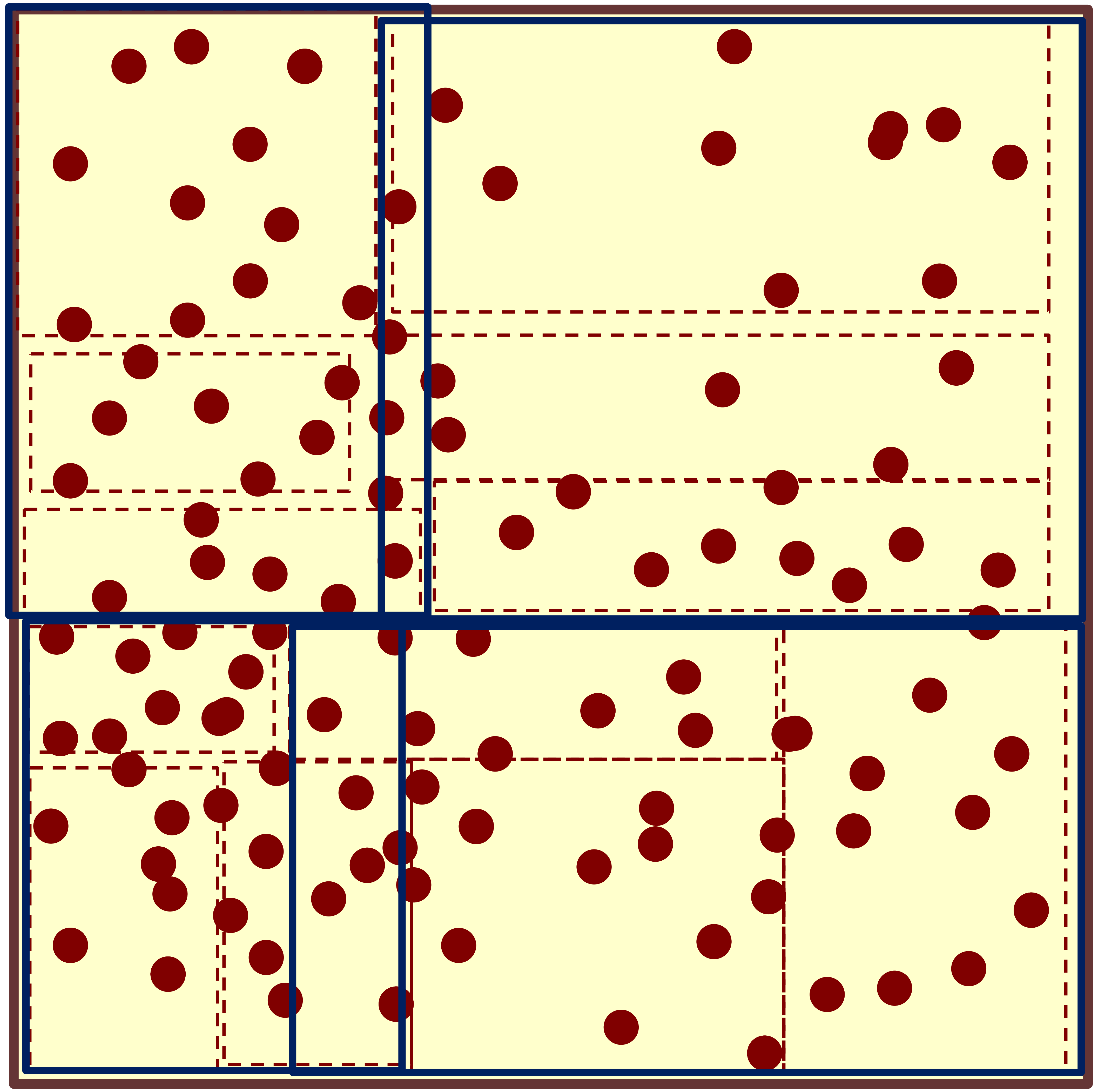
Geometry

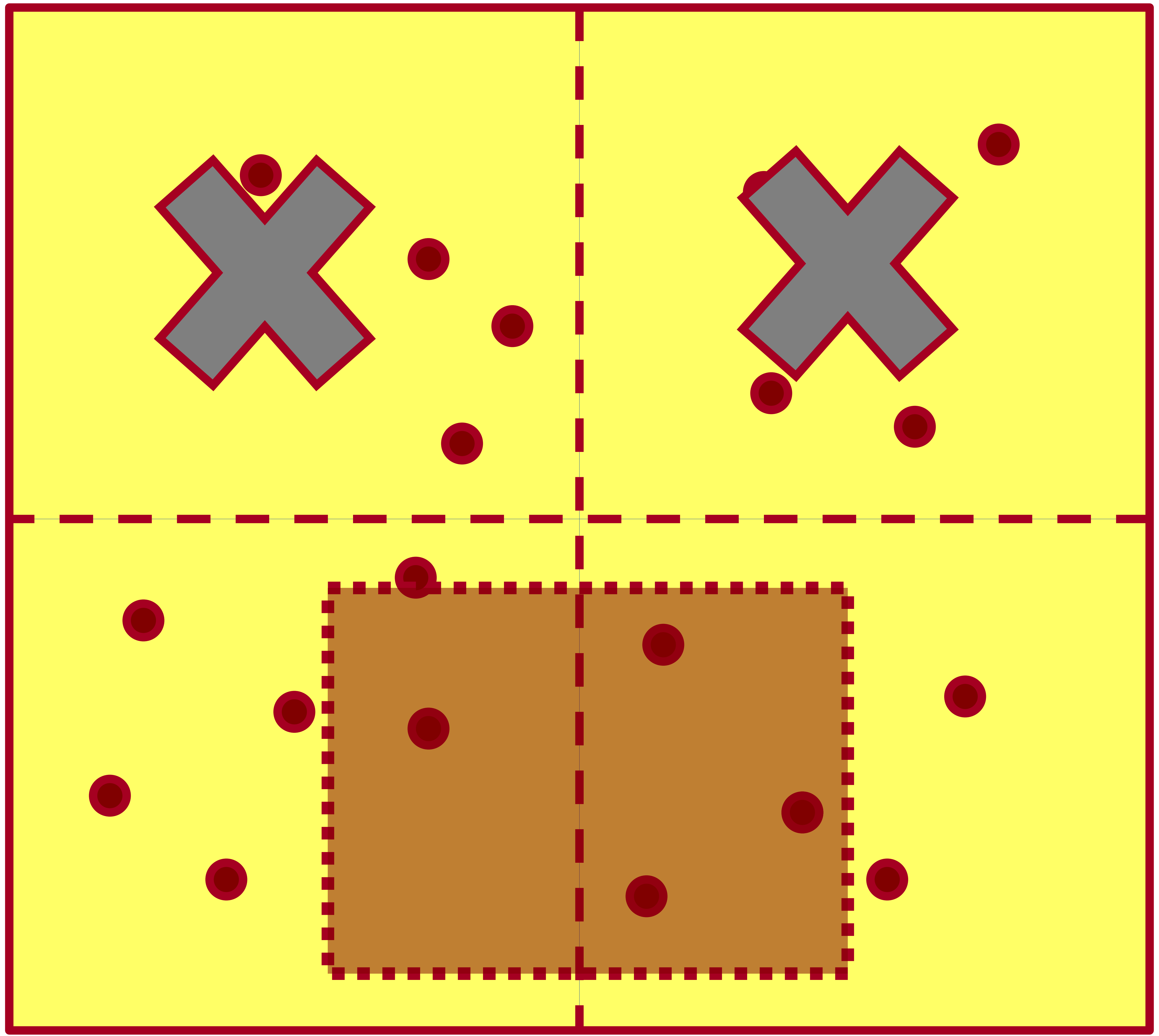
Washington
Oregon
Idaho
Nevada
Utah
California
Arizona

Feature

[A. Eldawy, 2021]

Beast Partitioning/Indexing & Range Query

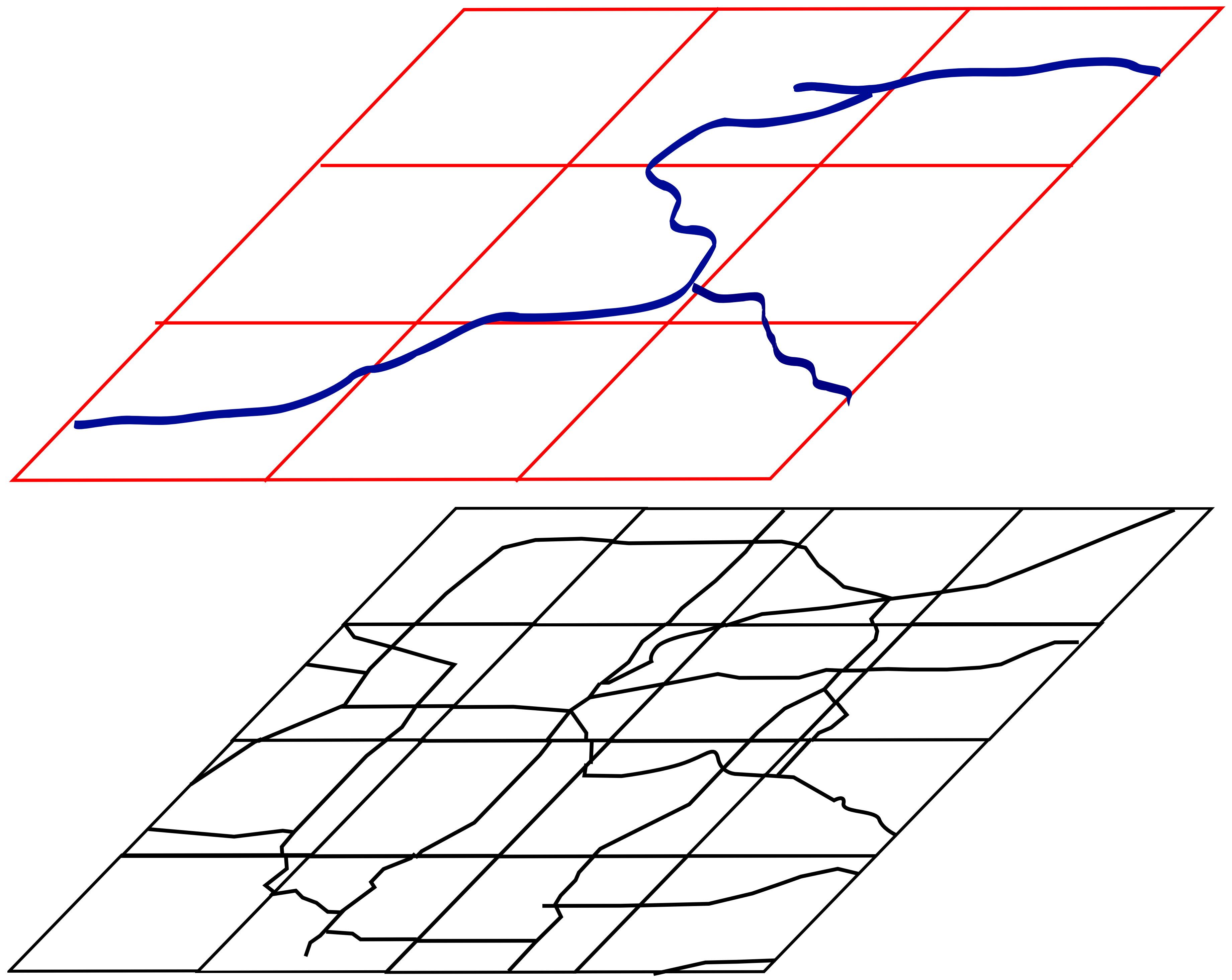




[A. Eldawy, 2021]

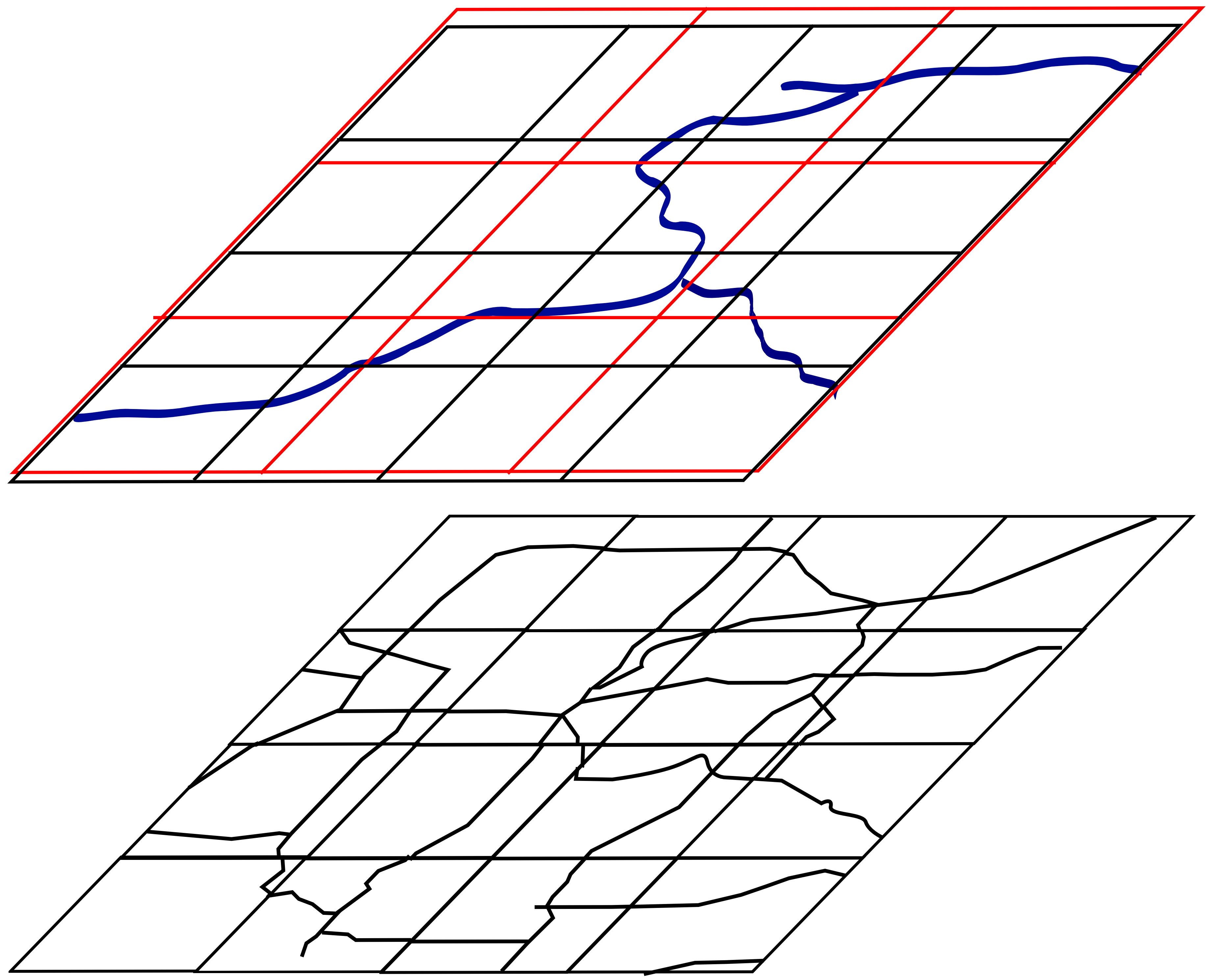
Beast Spatial Join

Join Directly



Total of 36 overlapping pairs

Partition – Join



Only 16 overlapping pairs

[A. Eldawy, 2021]