VAST · INFOVIS · SCIVIS
BIOVIS · LDAV

VIS 2013

Nanocubes for Real-Time Exploration of Spatiotemporal Datasets

Lauro Lins, James T. Klosowski, Carlos Scheidegger
AT&T Labs Research

http://nanocubes.net

We are interested in phenomena

with a spatiotemporal component

tweets

latitude	longitude	device	time
23.4	-40.3	iPhone	Aug 3, 2011 10:00
31.2	-41.3	Android	Aug 3, 2011 10:05
27.8	-39.3	iPhone	Aug 3, 2011 10:07
...	...	...	...
26.1	-38.1	Android	Jun 8, 2012 21:03
27.2	-44.3	Android	Jun 8, 2012 21:04
24.2	-39.7	iPhone	Jun 8, 2012 21:10

We are interested in phenomena

with a spatiotemporal component

tweets

How do we make sense of this...

latitude	longitude	device	time
23.4	-40.3	iPhone	Aug 3, 2011 10:00
31.2	-41.3	Android	Aug 3, 2011 10:05
27.8	-39.3	iPhone	Aug 3, 2011 10:07
...	...	...	...
26.1	-38.1	Android	Jun 8, 2012 21:03
27.2	-44.3	Android	Jun 8, 2012 21:04
24.2	-39.7	iPhone	Jun 8, 2012 21:

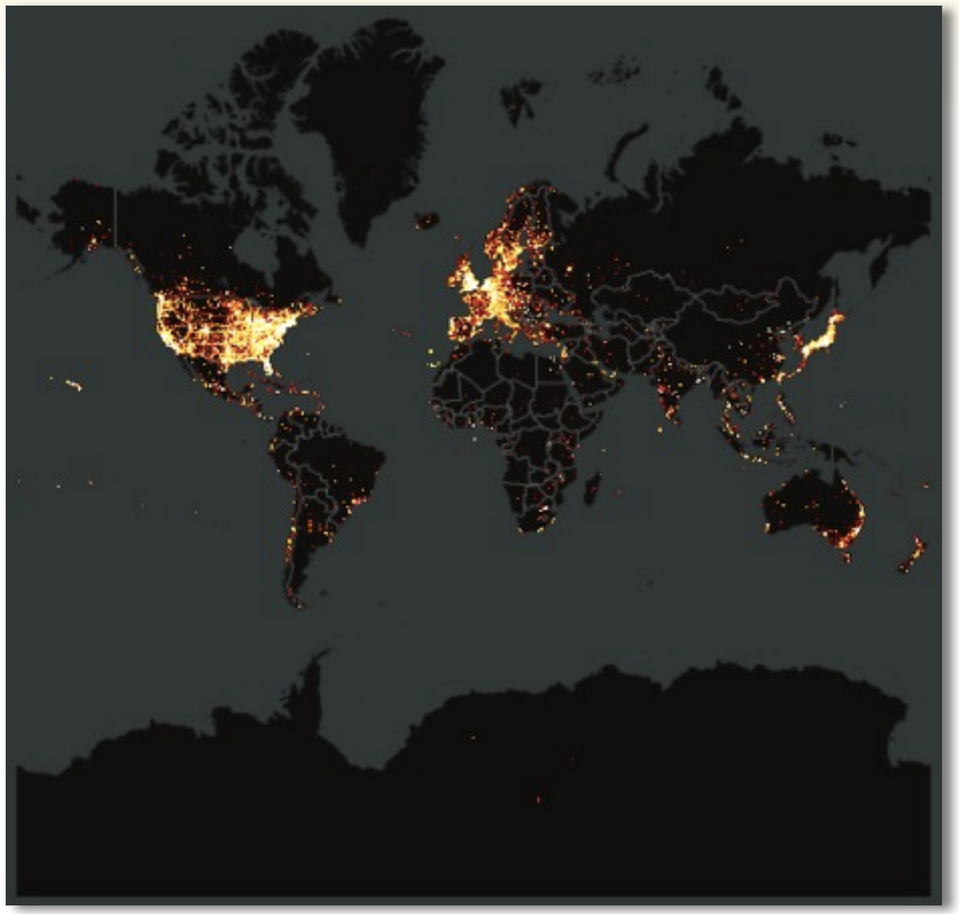
when there's a billion of these?

Demo

Problem Statement

How can we build interactive visualization interfaces when spatiotemporal datasets are large?

Query: produce a count heatmap of the world for all points in my dataset



Result

Query: produce a count heatmap of the world for all points in my database

Result

latitude	longitude	device	time
23.4	-40.3	iPhone	Aug 3, 2011 10:00
31.2	-41.3	Android	Aug 3, 2011 10:05
27.8	-39.3	iPhone	Aug 3, 2011 10:07
...	...	...	...
26.1	-38.1	Android	Jun 8, 2012 21:03
27.2	-44.3	Android	Jun 8, 2012 21:04
24.2	-39.7	iPhone	Jun 8, 2012 21:10

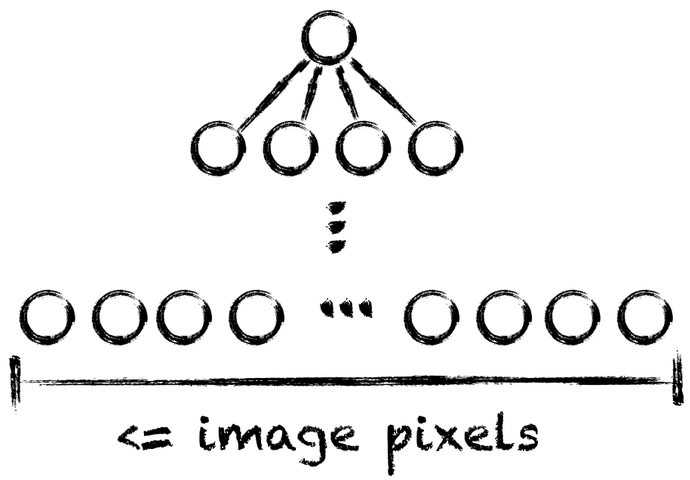
In if no aggregation was pre-computed then this query is proportional to "n"

Query: produce a count heatmap of the world for all points in my database

Result

latitude	longitude	device	time
23.4	-40.3	iPhone	Aug 3, 2011 10:00
31.2	-41.3	Android	Aug 3, 2011 10:05
27.8	-39.3	iPhone	Aug 3, 2011 10:07
...	...	...	...
26.1	-38.1	Android	Jun 8, 2012 21:03
27.2	-44.3	Android	Jun 8, 2012 21:04
24.2	-39.7	iPhone	Jun 8, 2012 21:10

if no aggregation was pre-computed then this query is proportional to "n"



if we pre-aggregate counts (e.g. quadtree) the query time becomes proportional to the number of reported pixels

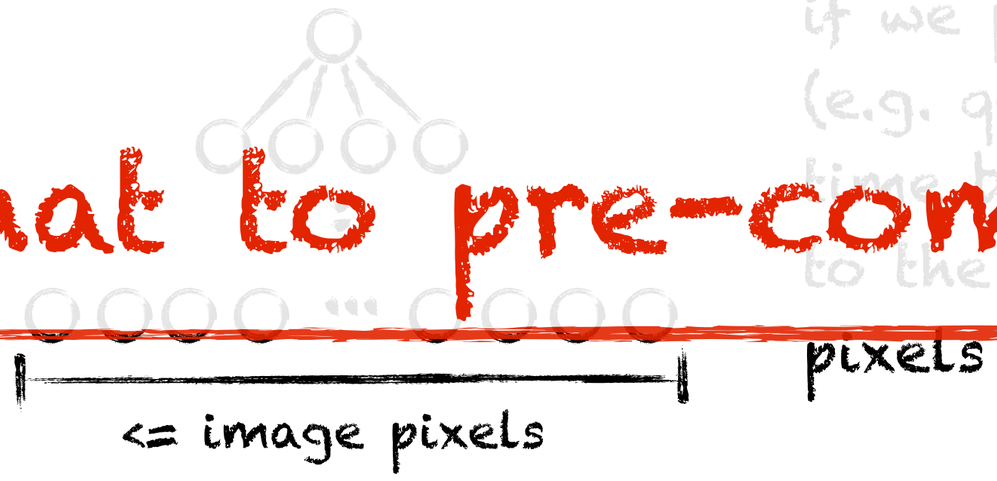
Query: produce a count heatmap of the world for all points in my database

latitude	longitude	device	time

What if data has time dimension? and other categorical variables?

Result

What to pre-compute?



Data Cubes!

[1987, J.Gray et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals]

... well known in Vis!

[2002, C. Stolte et al. Polaris]

[2003, C. Stolte et al. Multiscale visualization using data cubes]

[2007, J. Mackinlay et al. Show me]

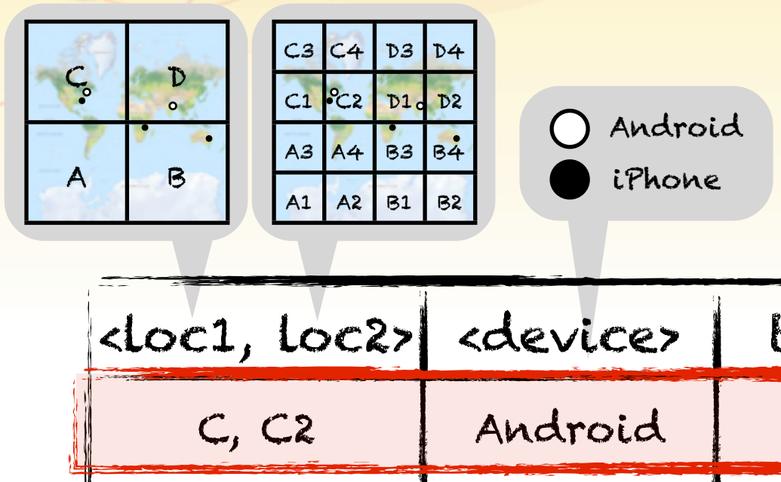
[2012, S. Kandel et al. Profiler]

[2013, Z. Liu et al. imMens]

What is a Nanocube?

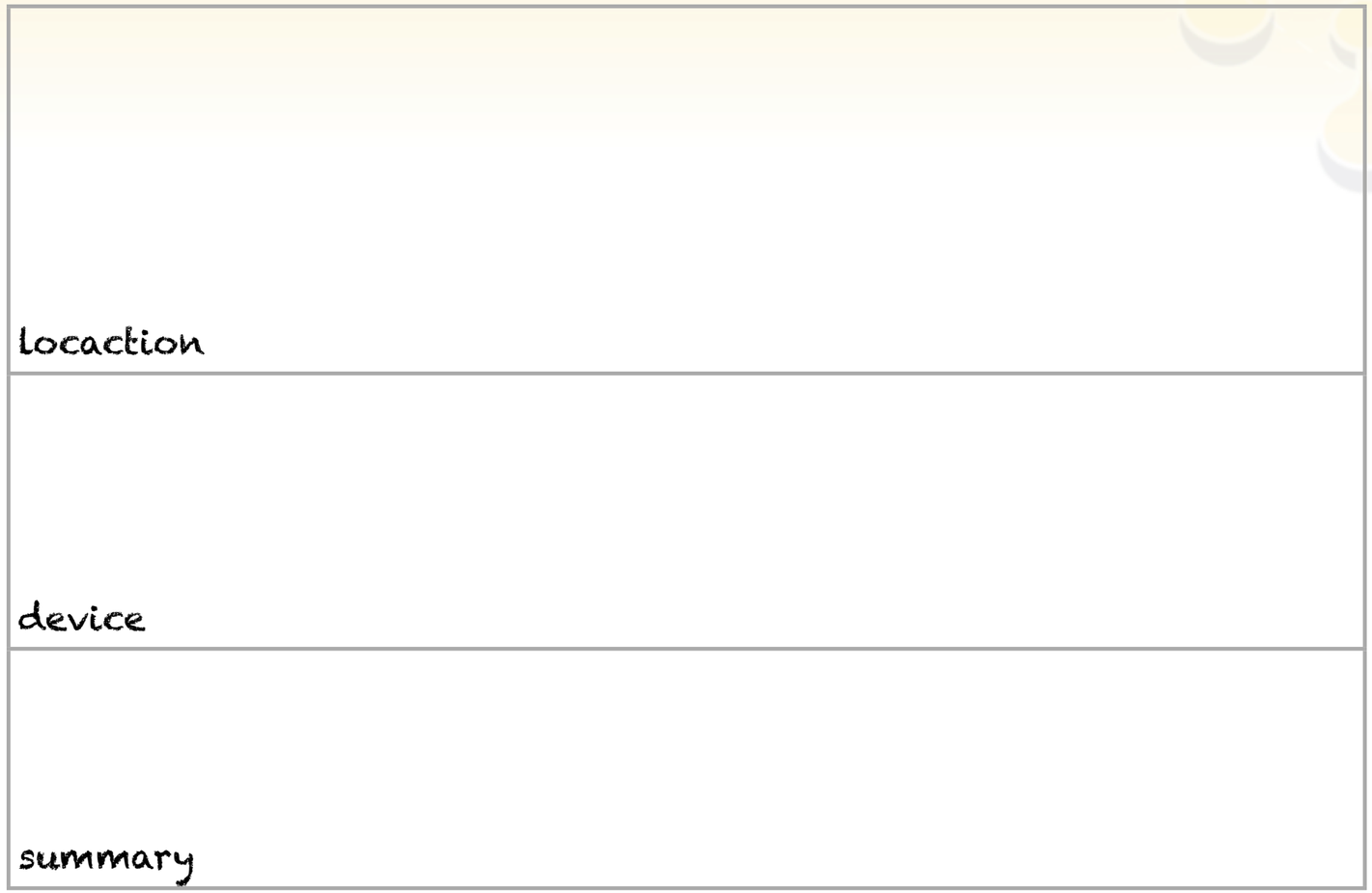
Nanocube is a Sparse in-Memory Data Cube of Spatiotemporal Events

Building a Nanocube

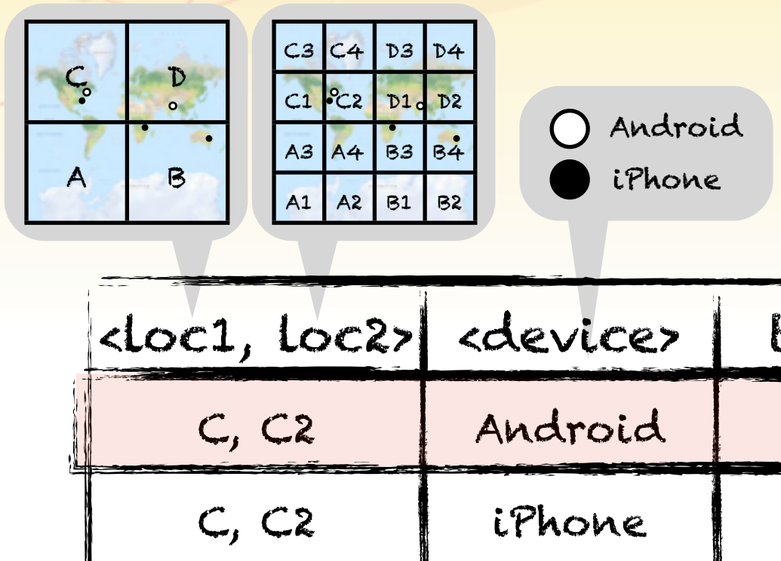


<loc1, loc2>	<device>	time
C, C2	Android	1
C, C2	iPhone	2
B, B3	iPhone	2
D, D1	Android	3
B, B4	iPhone	5

Insert (r_1)

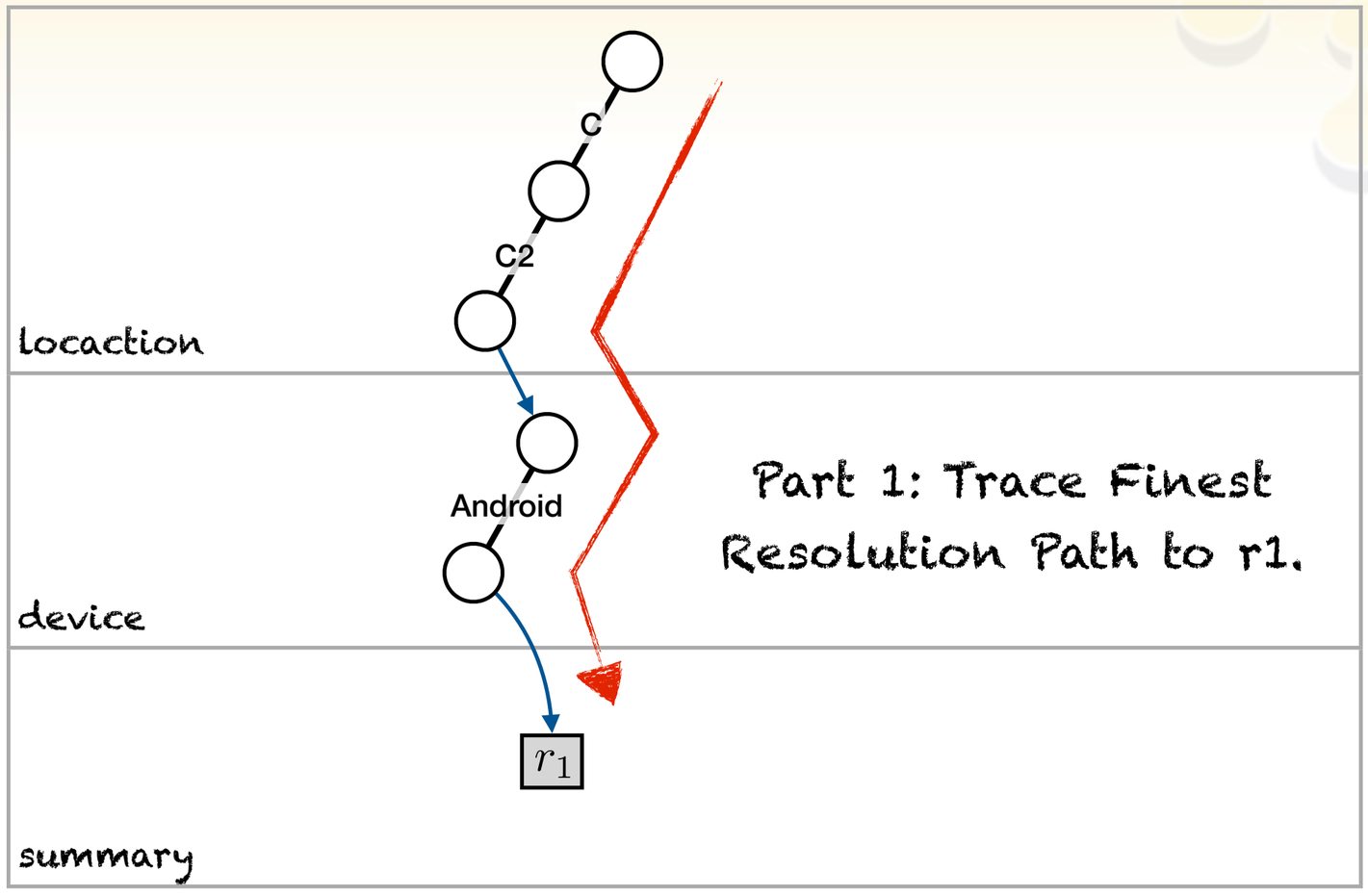


Building a Nanocube

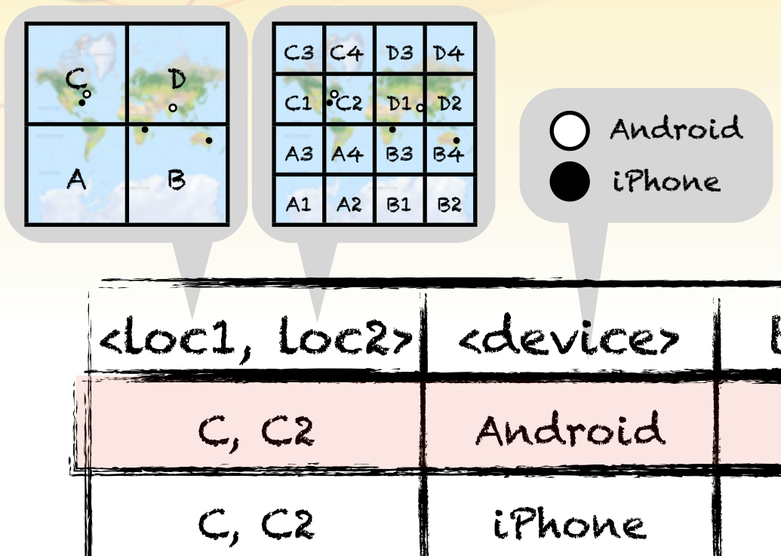


<loc1, loc2>	<device>	time
C, C2	Android	1
C, C2	iPhone	2
B, B3	iPhone	2
D, D1	Android	3
B, B4	iPhone	5

Insert (r_1)

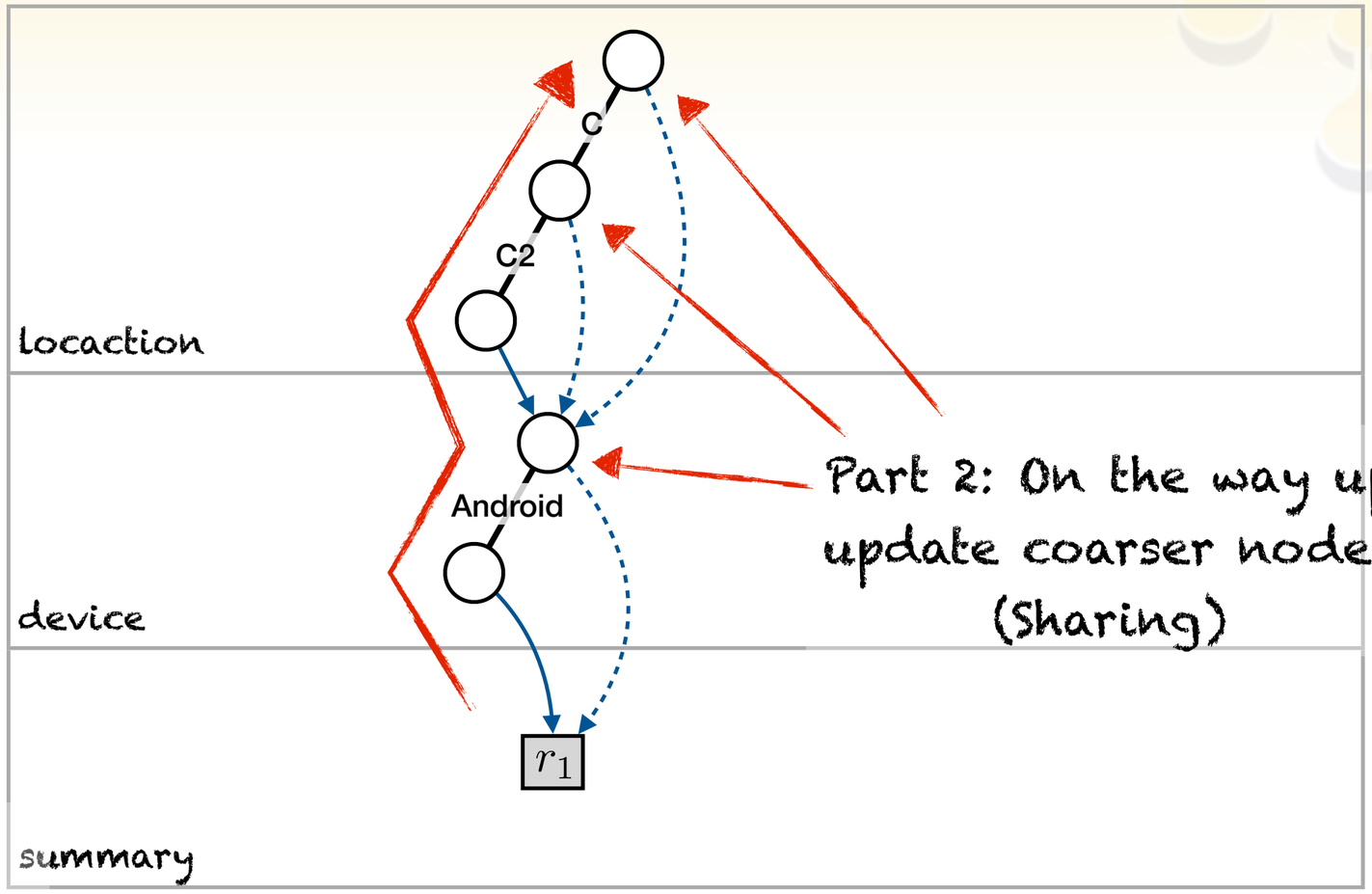


Building a Nanocube

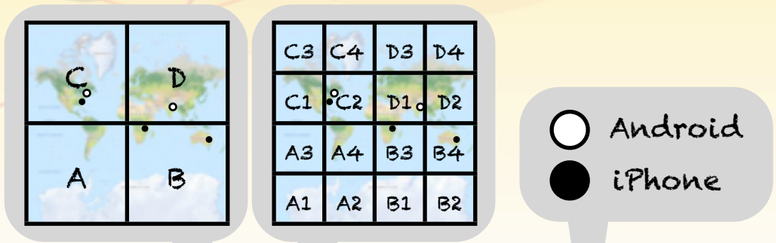


<loc1, loc2>	<device>	time
C, C2	Android	1
C, C2	iPhone	2
B, B3	iPhone	2
D, D1	Android	3
B, B4	iPhone	5

Insert (r_1)

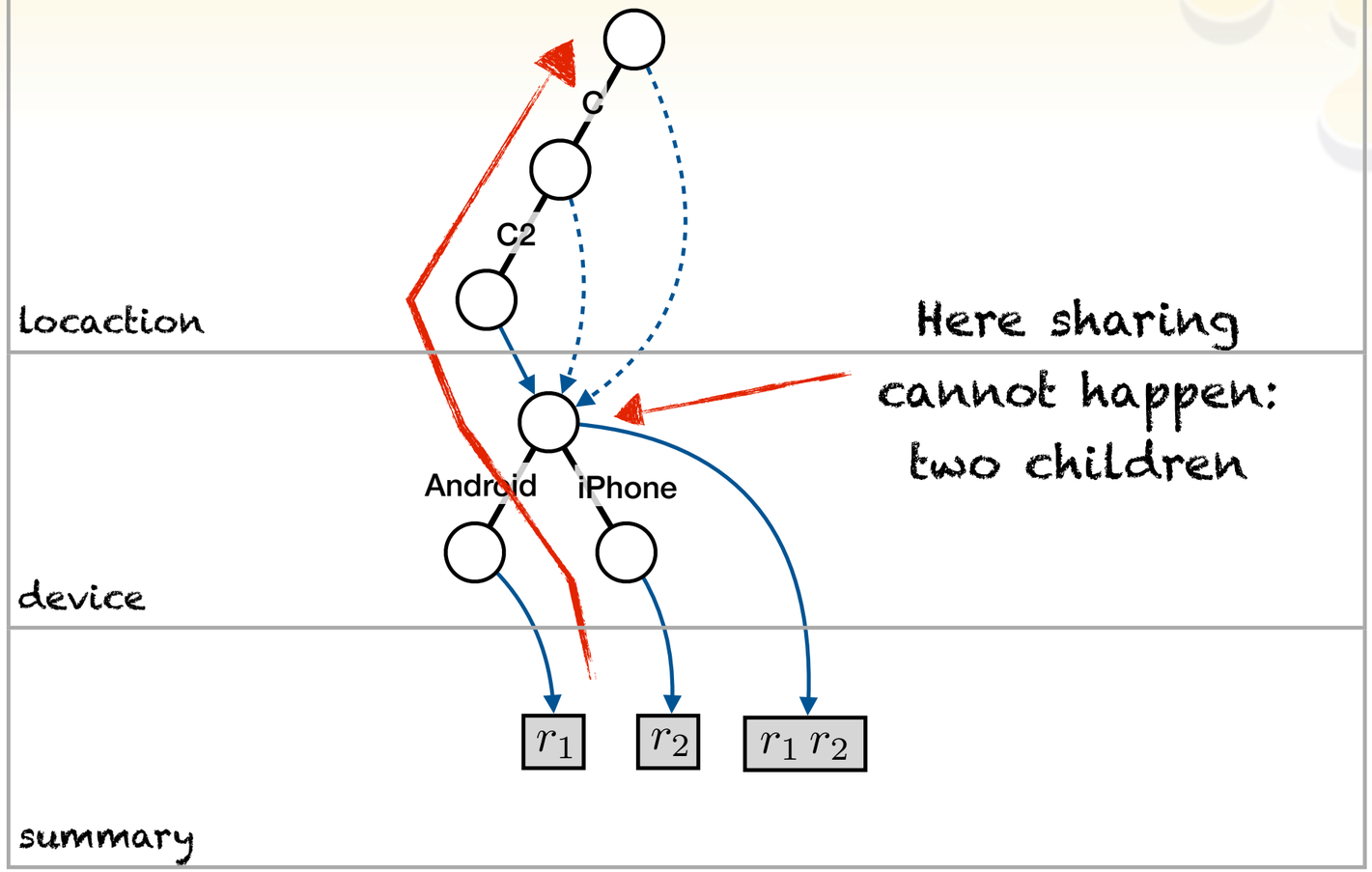


Building a Nanocube

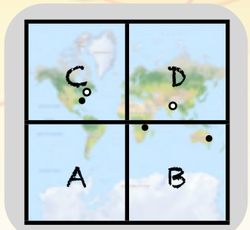


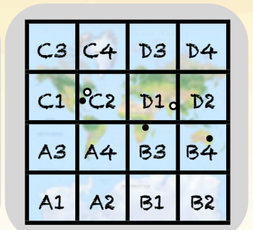
<loc1, loc2>	<device>	time
C, C2	Android	1
C, C2	iPhone	2
B, B3	iPhone	2
D, D1	Android	3
B, B4	iPhone	5

Insert (r_1) Insert (r_2)



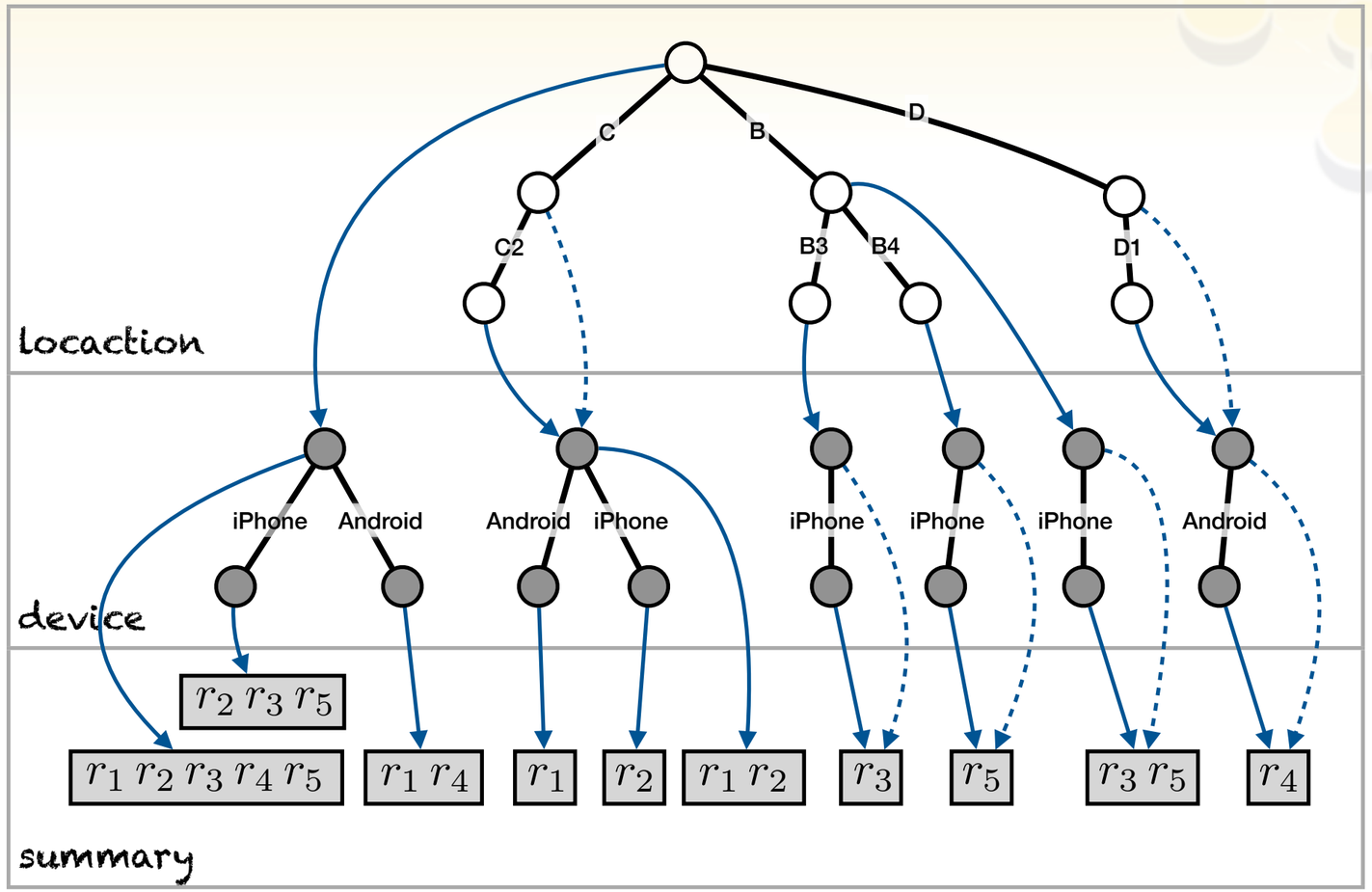
Building a Nanocube





<loc1, loc2>	<device>	time
C, C2	Android	1
C, C2	iPhone	2
B, B3	iPhone	2
D, D1	Android	3
B, B4	iPhone	5

Insert (r_1) Insert (r_2) ... Insert (r_5)

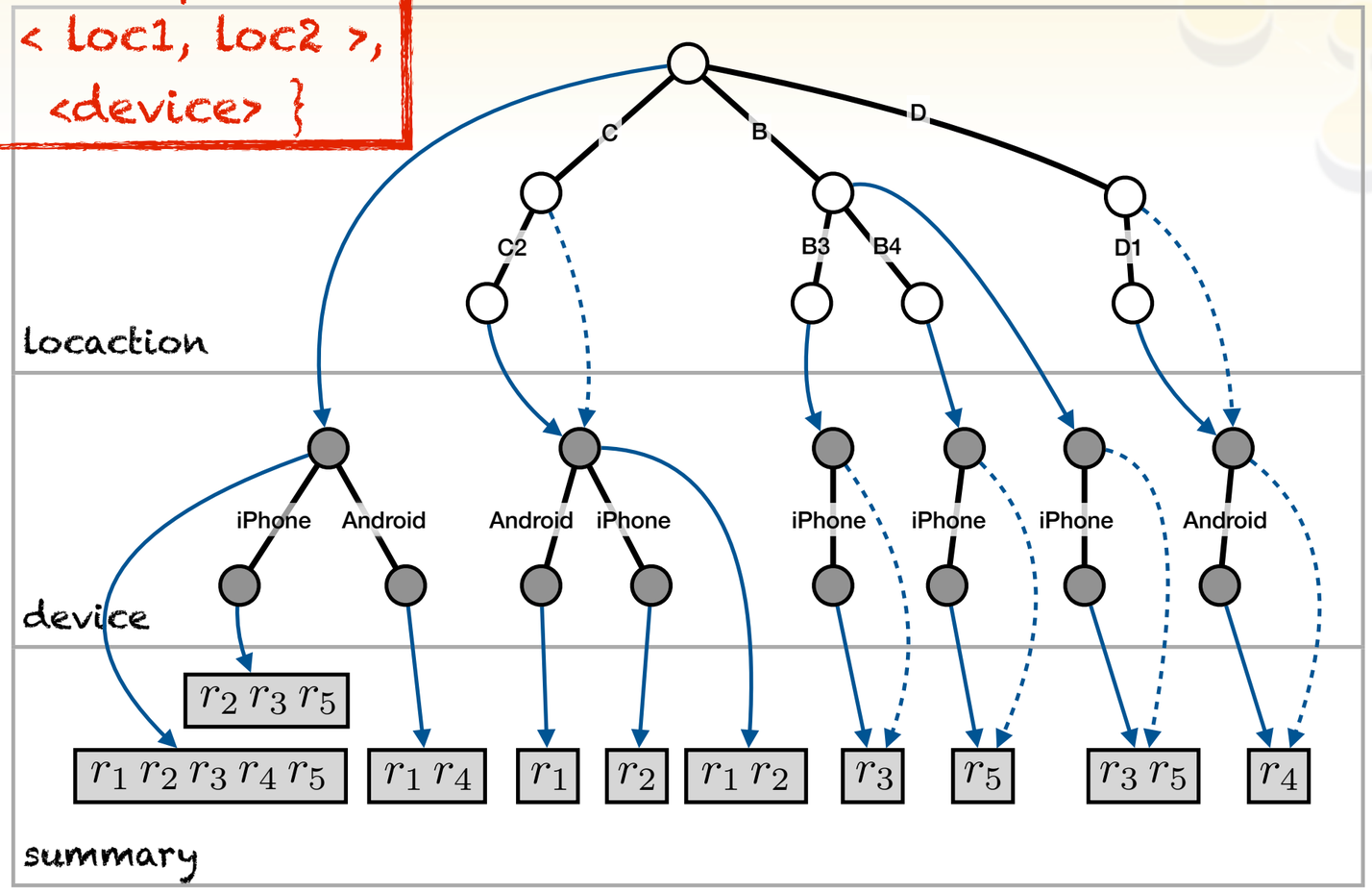


Building a Nanocube

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

Roll Up Cube

{ < loc1, loc2 >, <device> }

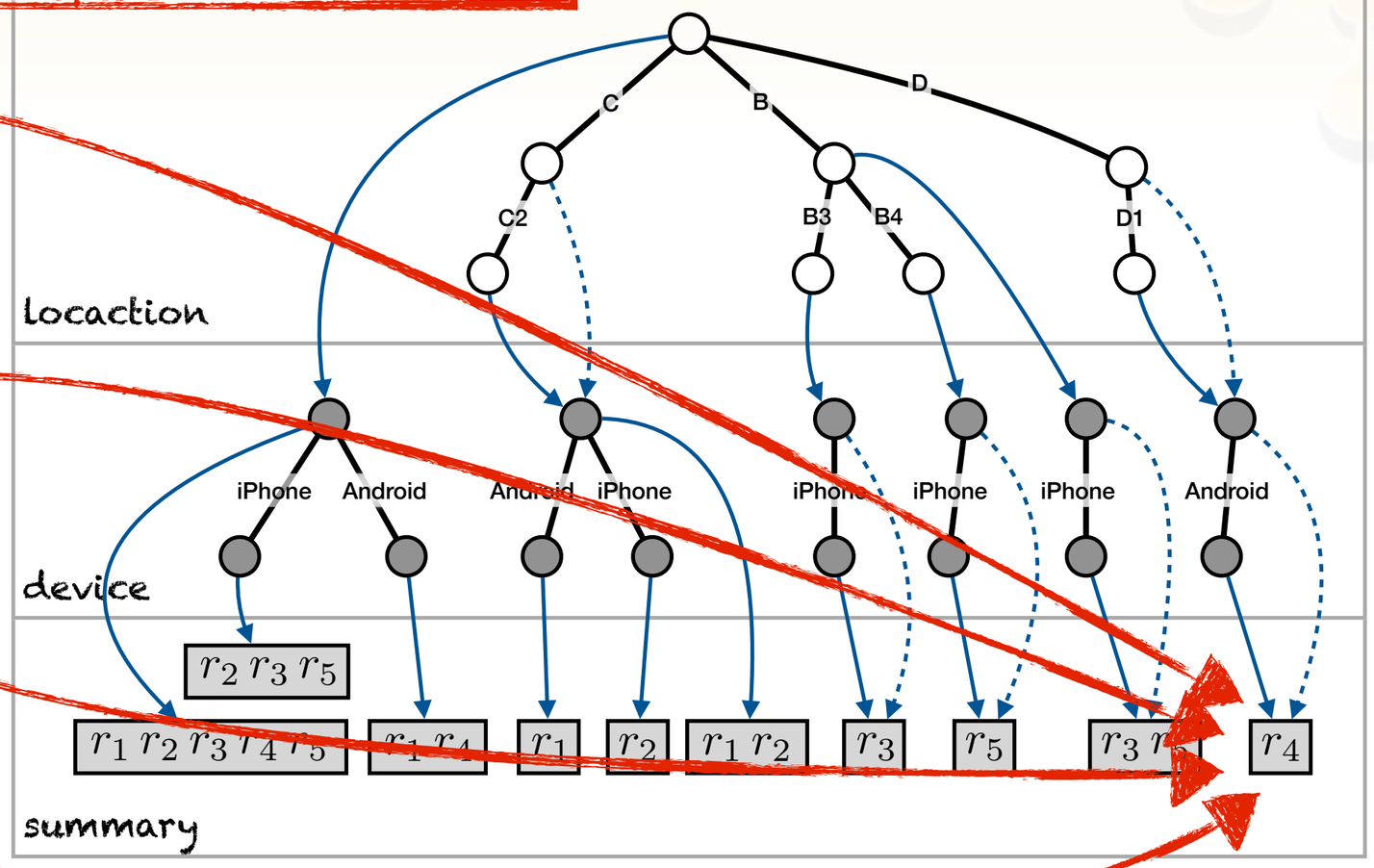


Building a Nanocube

Roll Up Cube
{ < loc1, loc2 >, <device> }

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

Optimal Encoding

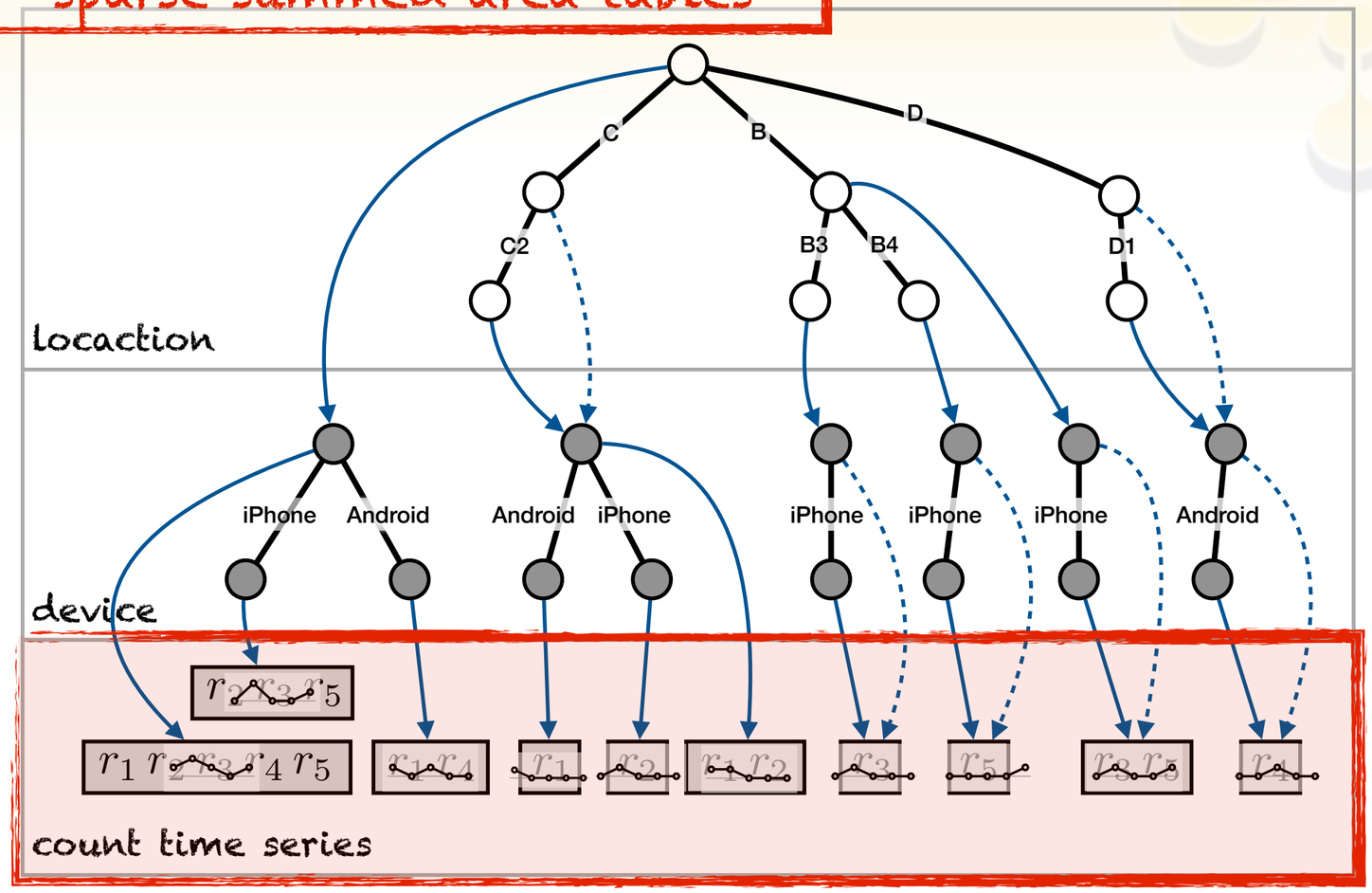


Building a Nanocube

Roll Up Cube
{ < loc1, loc2 >, <device> }

loc1	loc2	device	orig. records
*	*	*	
B	*	*	
C	*	*	
D	*	*	
*	*	Android	
*	*	iPhone	
C	C2	*	
B	B3	*	
B	B4	*	
D	D1	*	
C	*	Android	
C	*	iPhone	
B	*	iPhone	
D	*	Android	
C	C2	Android	
C	C2	iPhone	
B	B3	iPhone	
B	B4	iPhone	
D	D1	Android	

Time Series are encoded using sparse summed area tables

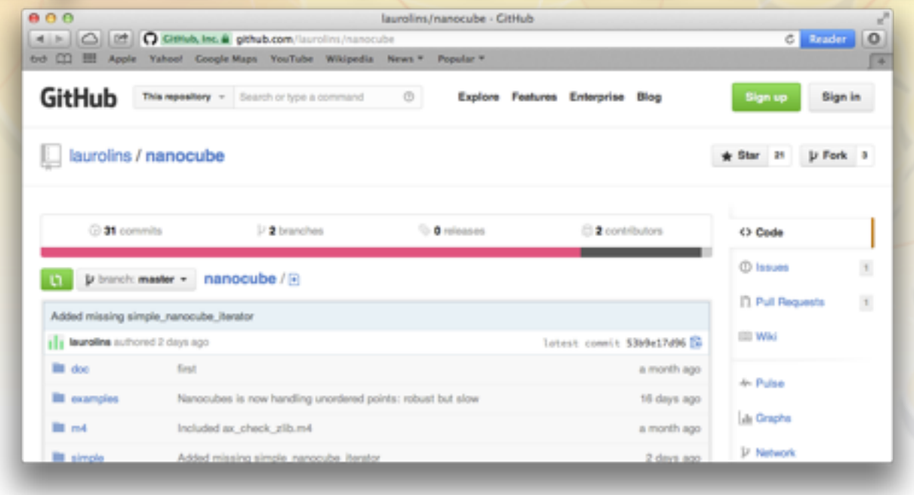


Implementation

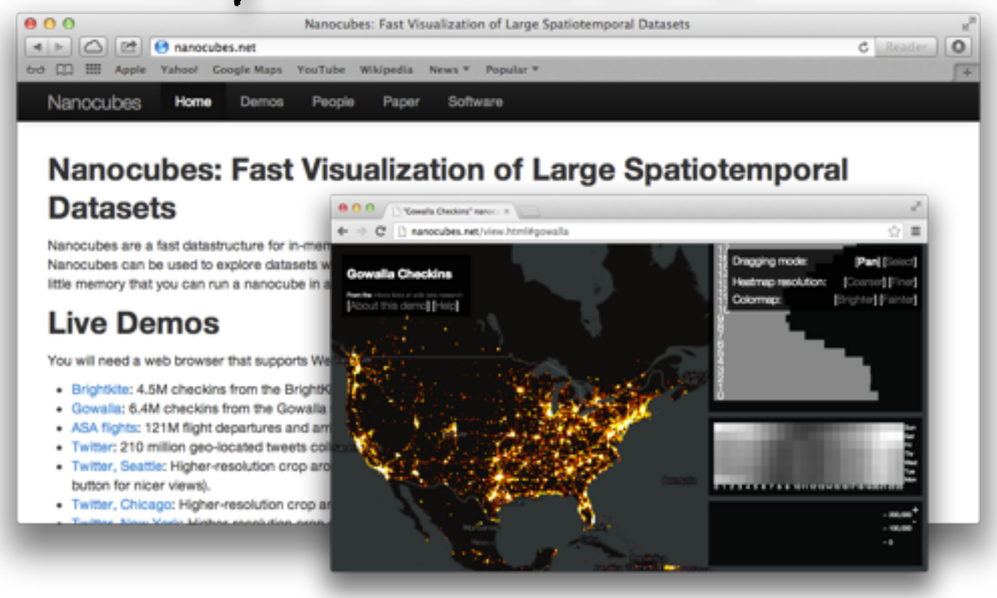
- Server - Back End
  - HTTP Server written in C++11
  - mongoose (http)
  - templates and boost::mpl
  - tagged pointers
  - **open source**

- Front Ends
  - C++11, OpenGL
  - Javascript, HTML5
    - WebGL, D3, Leaflet
      - (iPhones, iPads!)

http://github.com/laurolins/nanocube



http://nanocubes.net



Results

dataset	n	memory	time	keys	cardinality	schema
brightkite	4.5 M	1.6 GB	3.50 m	3.5 M	2^74	spatial(50), time(16), weekday(3), hour(5)
customer tix	7.8 M	2.5 GB	8.47 m	7.8 M	2^69	spatial(50), time(16), type(3)
flights	121 M	2.3 GB	31.13 m	43.3 M	2^75	spatial(50), time(16), carrier(5), delay(4)
twitter-small	210 M	10.2 GB	1.23 h	116.0 M	2^53	spatial(34), time(16), device(3)
twitter	210 M	46.4 GB	5.87 h	136.0 M	2^60	spatial(34), time(16), lang(5), device(3), app(2)
splom-10	1 B	4.3 MB	4.13 h	7.4 K	2^20	d1(4), d2(4), d3(4), d4(4), d5(4)
splom-50	1 B	166 MB	4.72 h	1.9 M	2^30	d1(6), d2(6), d3(6), d4(6), d5(6)
cdrs	1 B	3.6 GB	3.08 h	96.3 M	2^69	spatial(50), time(16), duration(3)

every nanocube
size <= 11GB RAM
except for one

Results

dataset	n	memory	time	keys	cardinality	schema
brightkite	4.5 M	1.6 GB	3.50 m	3.5 M	2^74	spatial(50), time(16), weekday(3), hour(5)
customer tix	7.8 M	2.5 GB	8.47 m	7.8 M	2^69	spatial(50), time(16), type(3)
flights	121 M	2.3 GB	31.13 m	43.3 M	2^75	spatial(50), time(16), carrier(5), delay(4)
twitter-small	210 M	10.2 GB	1.23 h	116.0 M	2^53	spatial(34), time(16), device(3)
twitter	210 M	46.4 GB	5.87 h	136.0 M	2^60	spatial(34), time(16), lang(5), device(3), app(2)
splom-10	1 B	4.3 MB	4.13 h	7.4 K	2^20	d1(4), d2(4), d3(4), d4(4), d5(4)
splom-50	1 B	166 MB	4.72 h	1.9 M	2^30	d1(6), d2(6), d3(6), d4(6), d5(6)
cdrs	1 B	3.6 GB	3.08 h	96.3 M	2^69	spatial(50), time(16), duration(3)

Limitations

- Summed-area table trick can only be used on the last dimension
- In the worst case, size of a nanocube grows exponentially with the number of new dimensions

Future and ongoing work

- Multiple spatial dimensions
- Nanocubes are insert-only
- Horizontal Scaling (Parallelism)
- Approximate Nanocubes

Conclusions

- Nanocubes provide spatiotemporal in-memory data cubes for large datasets
- they enable Interactive Visual Interfaces for datasets much larger than previously possible

- You can explore big data with your laptop, tablet and phone!

Acknowledgements:

Stephen North

Divesh Srivastava

Luciano Barbosa

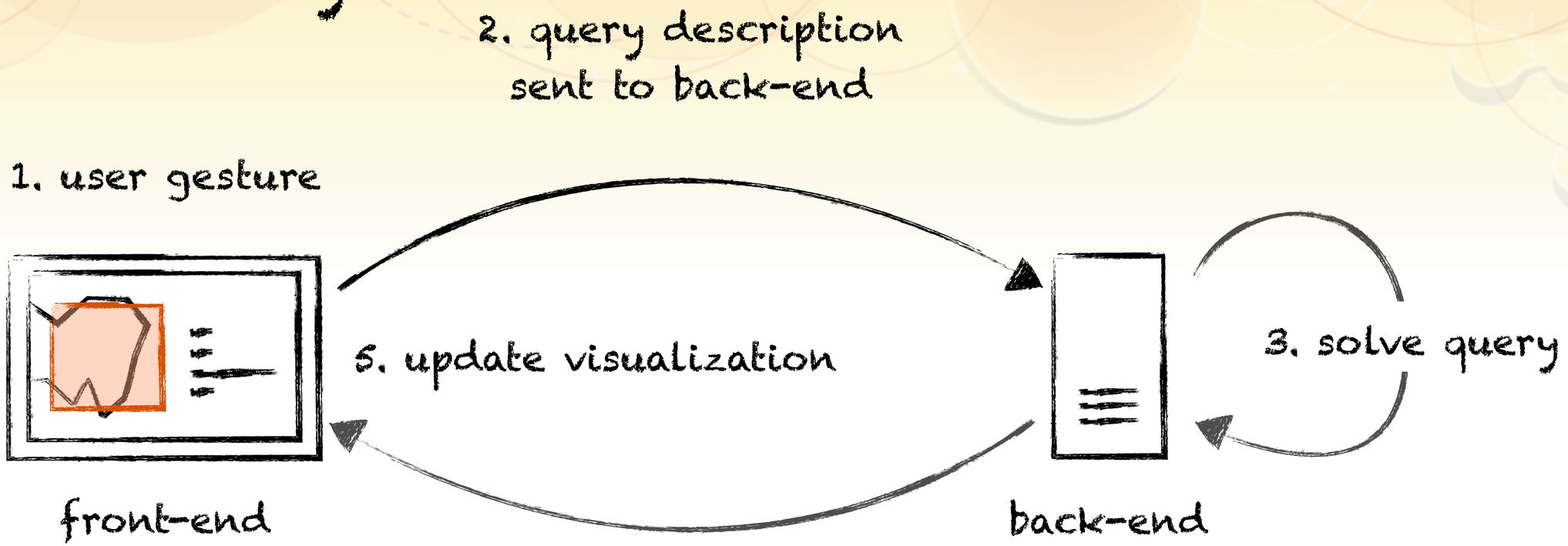
Thank you - questions?

http://nanocubes.net

llins@research.att.com

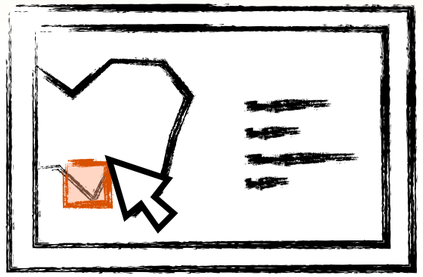
http://github.com/laurolins/nanocube

Interaction Cycle

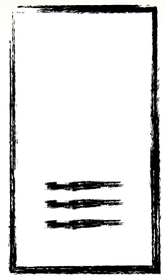


$$t2 + t3 + t4 + t5 <= \begin{cases} 33.3 \text{ ms} & (30 \text{ fps}) \\ 66.6 \text{ ms} & (15 \text{ fps}) \\ 100 \text{ ms} & (10 \text{ fps}) \end{cases}$$

Interaction Cycle



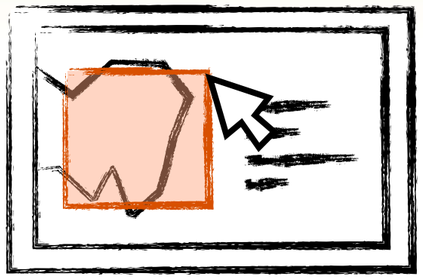
front-end



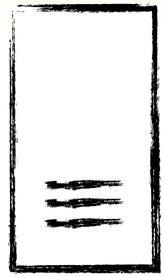
back-end

Interaction Cycle

1. user gesture



front-end



back-end

Interaction Cycle

2. query description sent to back-end

1. user gesture

front-end

back-end

Interaction Cycle

2. query description sent to back-end

1. user gesture

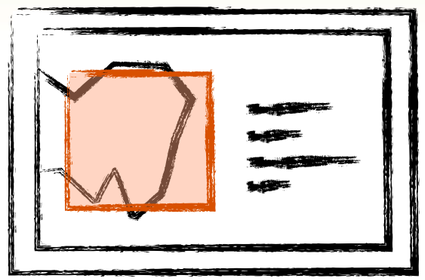
front-end

back-end

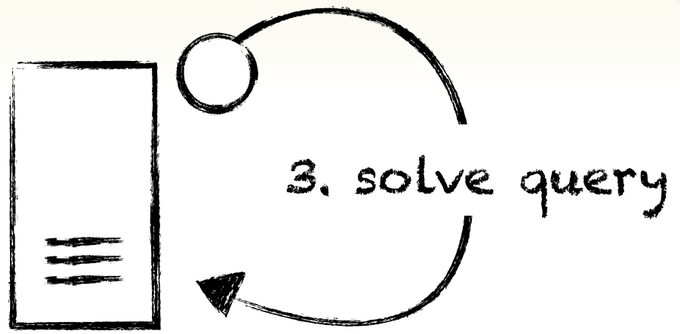
Interaction Cycle

2. query description
sent to back-end

1. user gesture



front-end

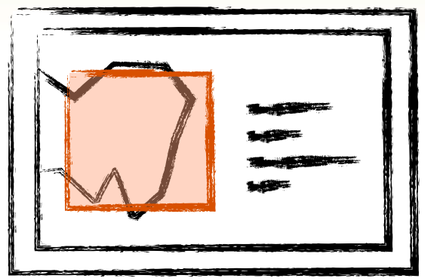


back-end

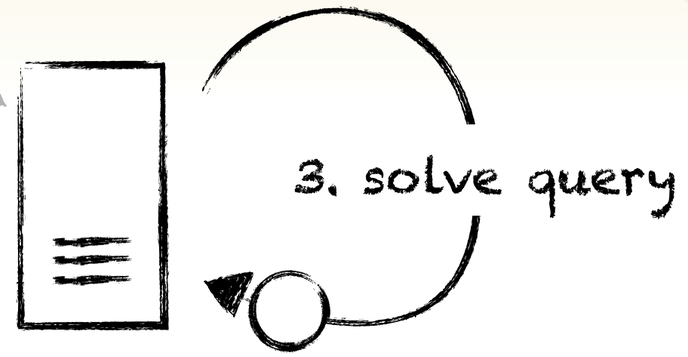
Interaction Cycle

2. query description
sent to back-end

1. user gesture



front-end



back-end

Interaction Cycle

2. query description
sent to back-end

1. user gesture

4. query result
sent to front-end

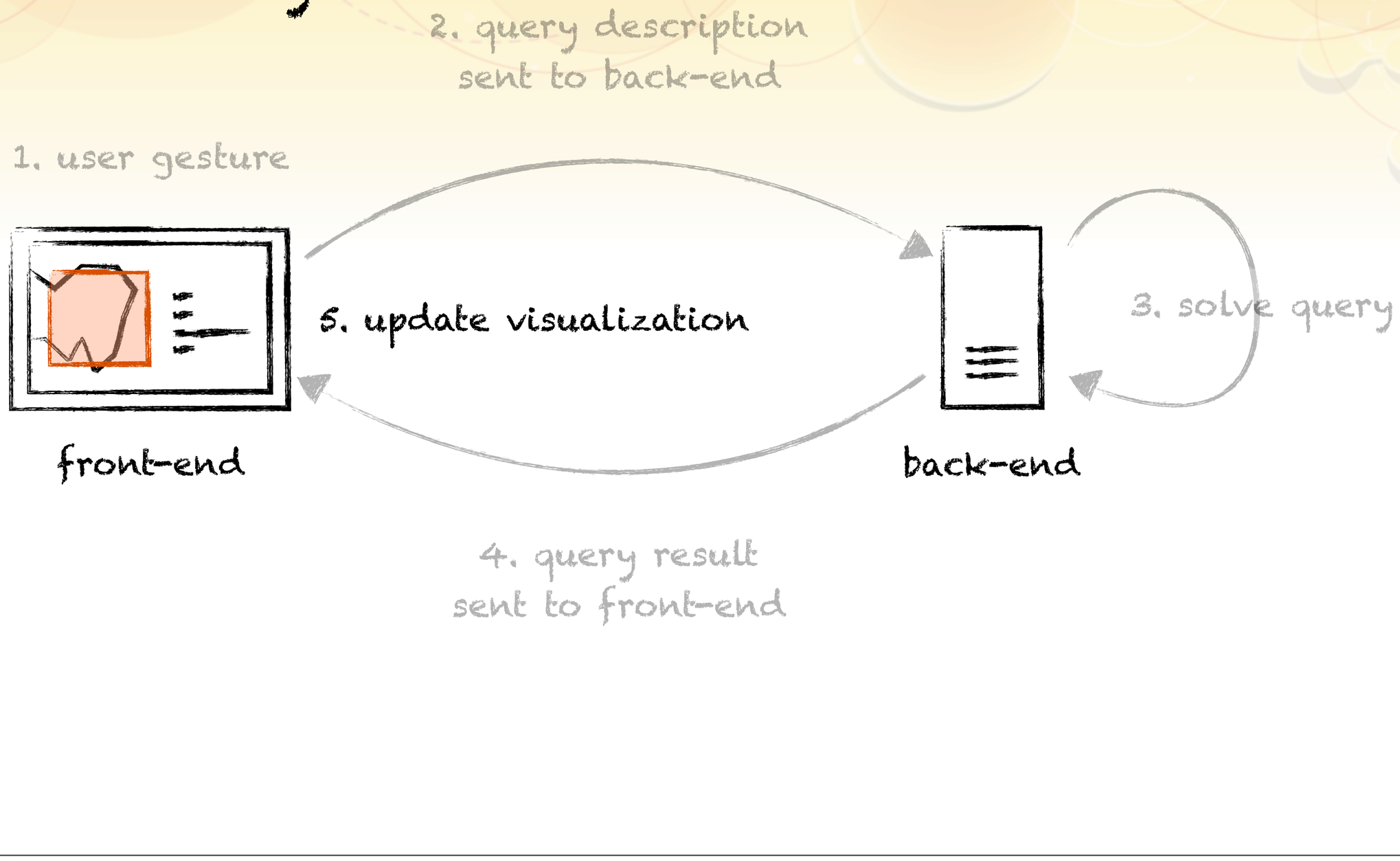
Interaction Cycle

2. query description
sent to back-end

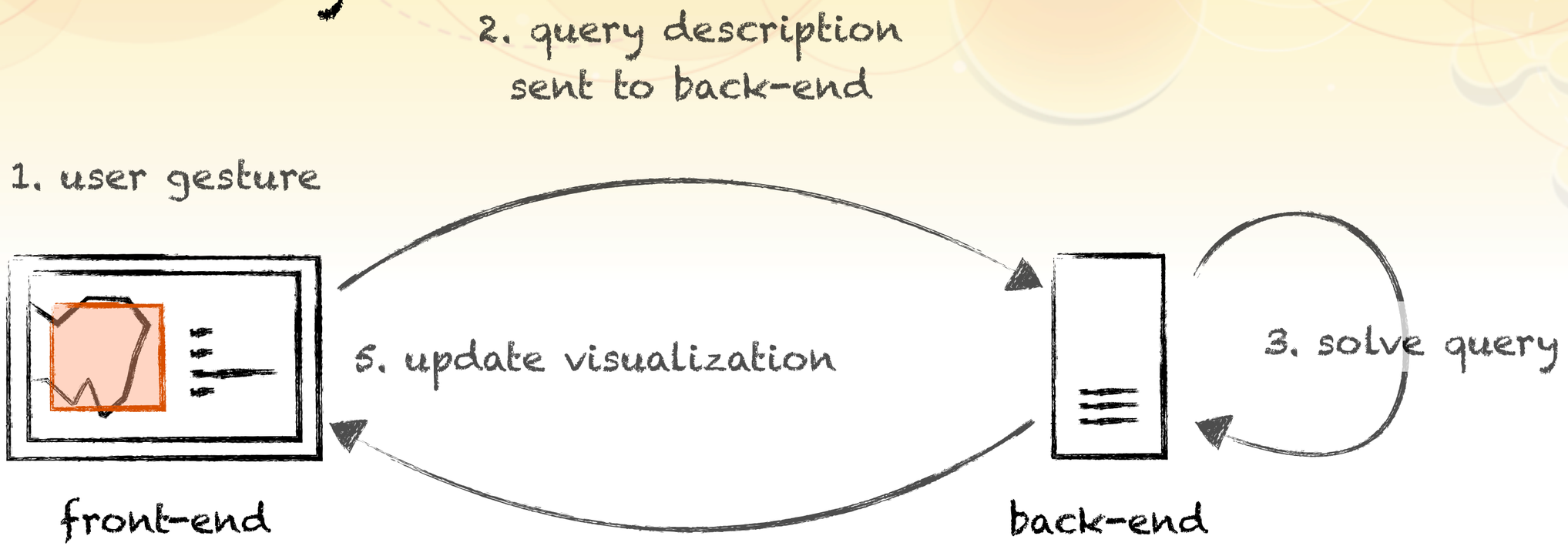
1. user gesture

4. query result
sent to front-end

Interaction Cycle

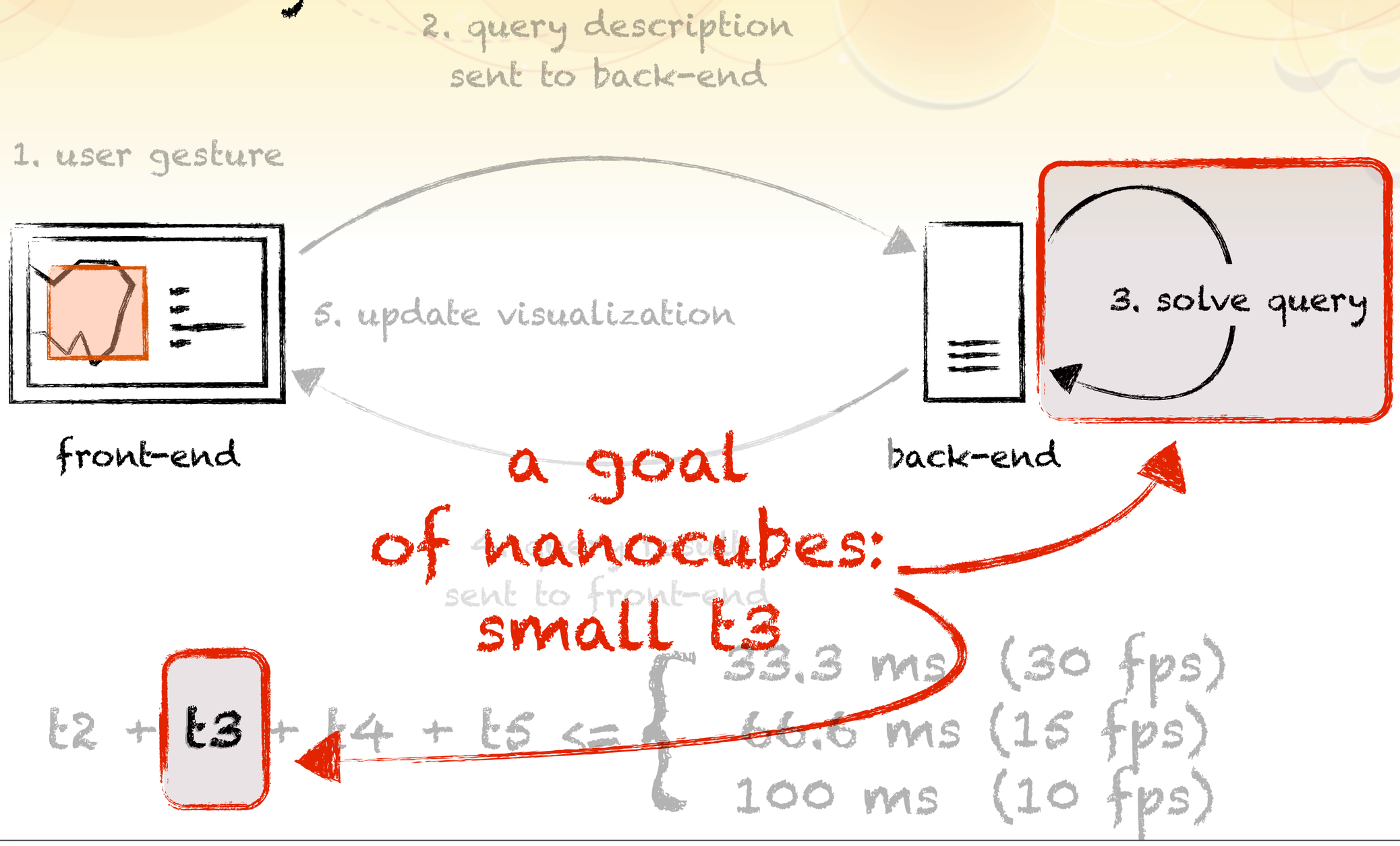


Interaction Cycle



$$t2 + t3 + t4 + t5 <= \begin{cases} 33.3 \text{ ms} & (30 \text{ fps}) \\ 66.6 \text{ ms} & (15 \text{ fps}) \\ 100 \text{ ms} & (10 \text{ fps}) \end{cases}$$

Interaction Cycle

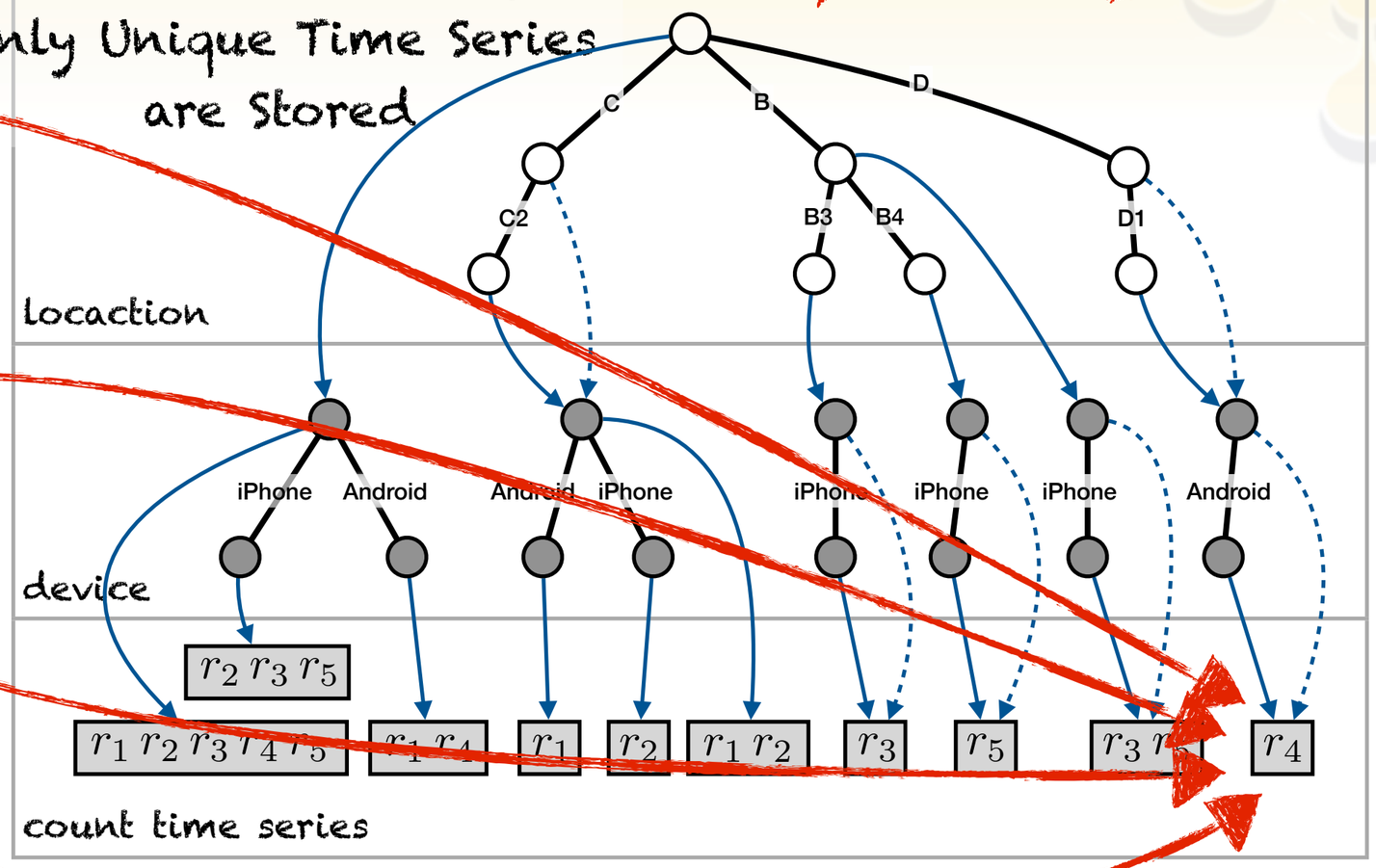


Building a Nanocube

[2002, 2003 Sismanis et al. Dwarf Cubes]

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

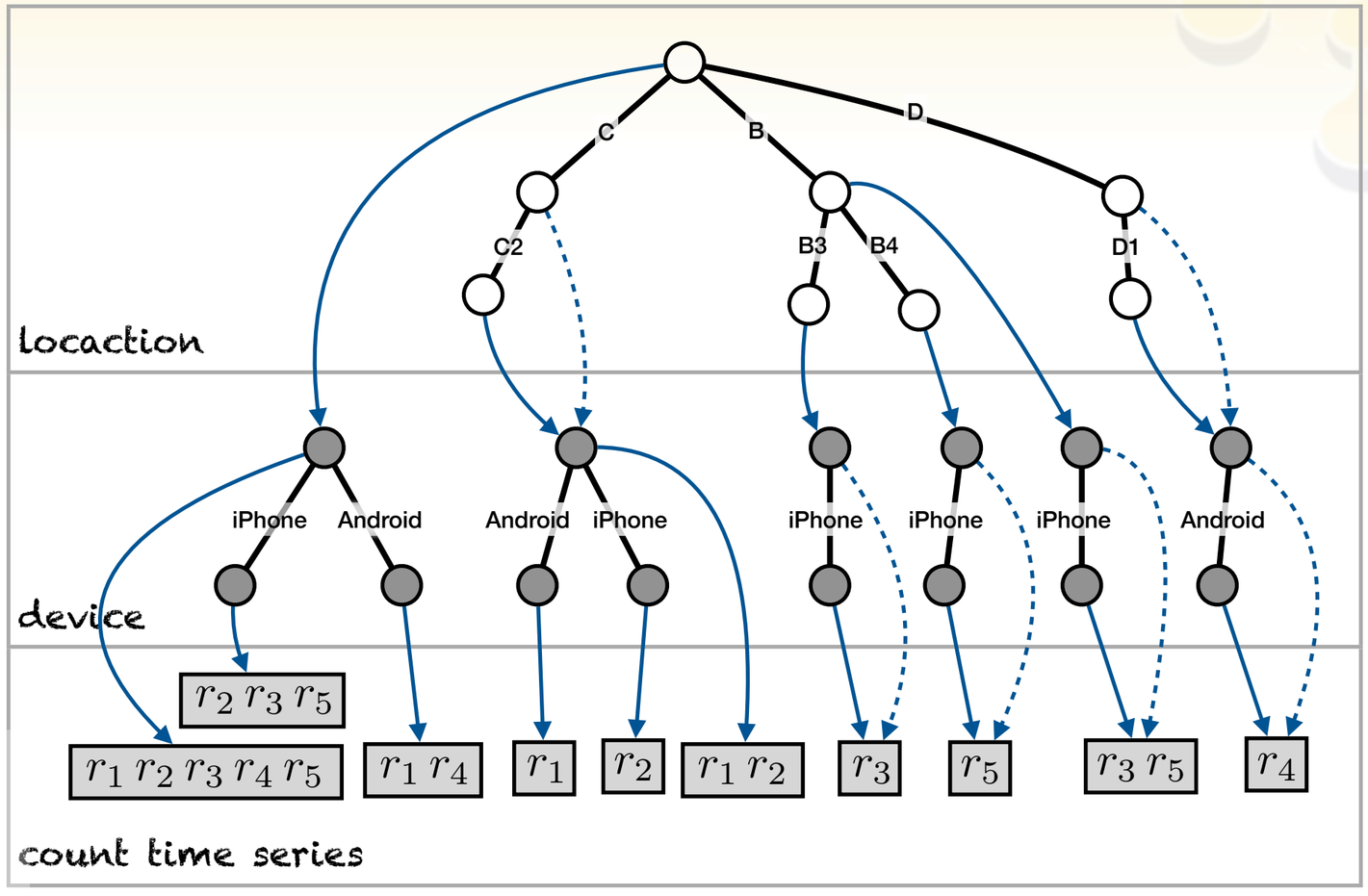
Efficient Encoding:
Only Unique Time Series are Stored



Building a Nanocube

loc1	loc2	device	time
C	C2	Android	1
C	C2	iPhone	2
B	B3	iPhone	3
D	D1	Android	4
B	B4	iPhone	5

Nanocube for Roll Up Cube
{ < loc1, loc2 >, <device> }
w/ Count Time Series



Nanocube Trick #1 for Memory Efficiency

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

Nanocube Trick #1 for Memory Efficiency

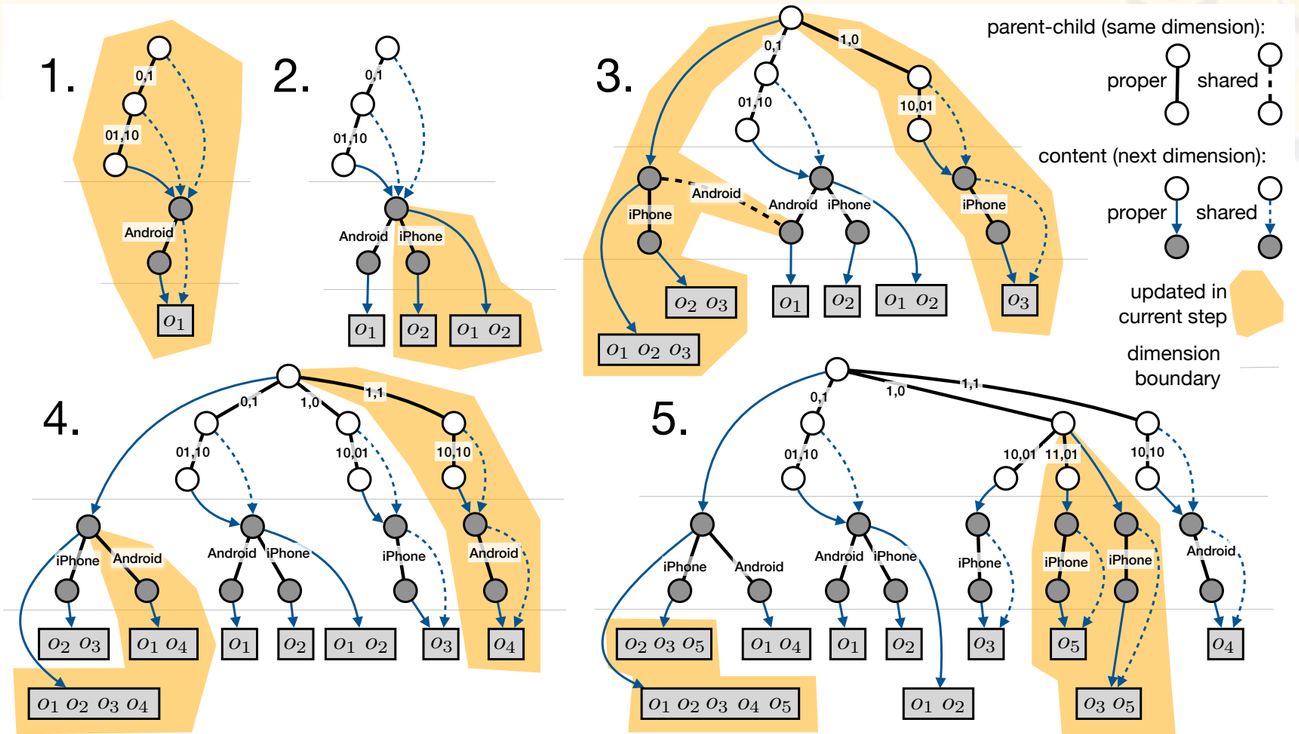
loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

Nanocube Trick #1 for Memory Efficiency

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

Nanocube's Encoding

loc1	loc2	device	time series
*	*	*	
B	*	*	
C	*	*	
D	*	*	
*	*	Android	
*	*	iPhone	
C	C2	*	
B	B3	*	
B	B4	*	
D	D1	*	
C	*	Android	
C	*	iPhone	
B	*	iPhone	
D	*	Android	
C	C2	Android	
C	C2	iPhone	
B	B3	iPhone	
B	B4	iPhone	
D	D1	Android	



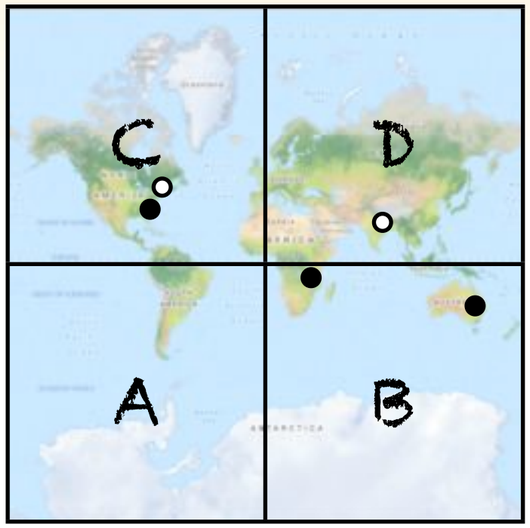
Nanocube Trick #1 for Memory Efficiency

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

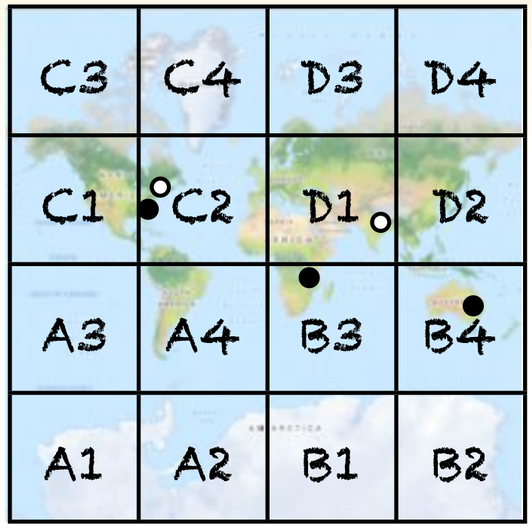
What to pre-compute? Data Cubes!

What to pre-compute? Data Cubes!

loc1



loc2



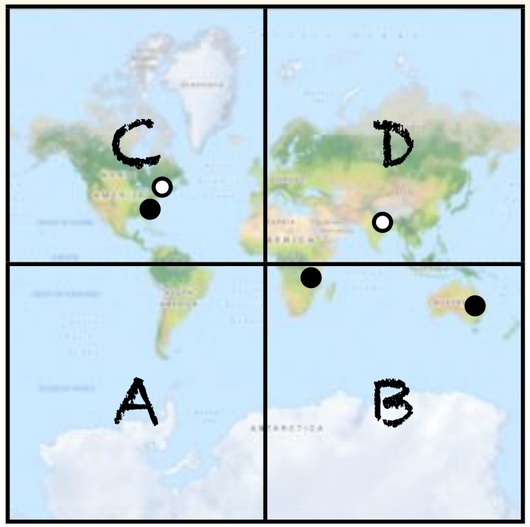
device:

○ Android

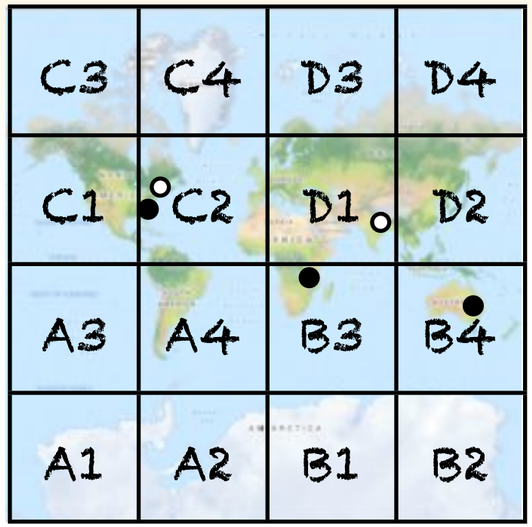
● iPhone

What to pre-compute? Data Cubes!

loc1



loc2



device:

○ Android

● iPhone

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

What to pre-compute? Data Cubes!

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

group by { device }

loc1	loc2	device	orig. rows
*	*	Android	{ r1, r4 }
*	*	iPhone	{ r2, r3, r5 }

What to pre-compute? Data Cubes!

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

group by { loc1, device }

loc1	loc2	device	orig. rows
C	*	Android	{ r1 }
C	*	iPhone	{ r2 }
B	*	iPhone	{ r3, r5 }
D	*	Android	{ r4 }

What to pre-compute? Data Cubes!

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

cube { loc1, loc2, device }

= group by {} +
group by { loc1 } +
group by { loc2 } +
group by { device } +
group by { loc1, loc2 } +
group by { loc1, device } +
group by { loc2, device } +
group by { loc1, loc2, device }

What to pre-compute? Data Cubes!

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B4	iPhone

relation

By construction these two cases are redundant

cube { loc1, loc2, device }

= group by {} +
group by { loc1 } +
group by { loc2 } +
group by { device } +
group by { loc1, loc2 } +
group by { loc1, device } +
group by { loc2, device } +
group by { loc1, loc2, device }

What to pre-compute? Data Cubes!

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

cube { loc1, loc2, device }

= group by {} +
group by { loc1 } +
group by { device } +
group by { loc1, device } +
group by { loc2, device } +
group by { loc1, loc2 } +
group by { loc1, loc2, device }

By construction these two cases are redundant

group by { loc2 }

group by { loc1, loc2 }

Roll Up on < loc1, loc2 > = group by {} +
group by {loc1} + group by {loc1, loc2 }

What to pre-compute? Data Cubes!

Roll Up Cube { < loc1, loc2 >, <device> }

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

= group by {} +
group by { loc1 } +
~~group by { loc2 } +~~
group by { device } +
group by { loc1, loc2 } +
group by { loc1, device } +
~~group by { loc2, device } +~~
group by { loc1, loc2, device }

What to pre-compute? Data Cubes!

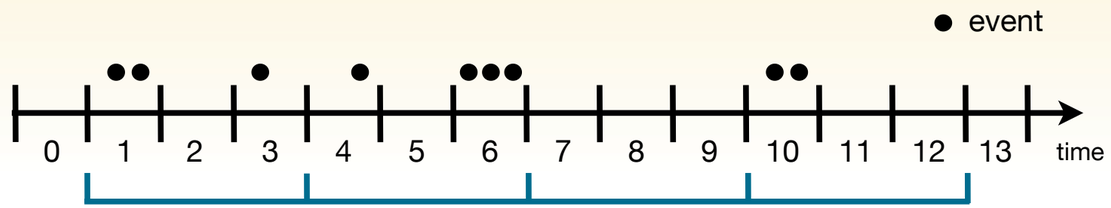
Roll Up Cube { < loc1, loc2 >, <device> }

loc1	loc2	device
C	C2	Android
C	C2	iPhone
B	B3	iPhone
D	D1	Android
B	B4	iPhone

relation

loc1	loc2	device	orig. records
*	*	*	{r1,r2,r3,r4,r5}
B	*	*	{r3,r5}
C	*	*	{r1,r2}
D	*	*	{r4}
*	*	Android	{r1,r4}
*	*	iPhone	{r2,r3,r5}
C	C2	*	{r1,r2}
B	B3	*	{r3}
B	B4	*	{r5}
D	D1	*	{r4}
C	*	Android	{r1}
C	*	iPhone	{r2}
B	*	iPhone	{r3,r5}
D	*	Android	{r4}
C	C2	Android	{r1}
C	C2	iPhone	{r2}
B	B3	iPhone	{r3}
B	B4	iPhone	{r5}
D	D1	Android	{r4}

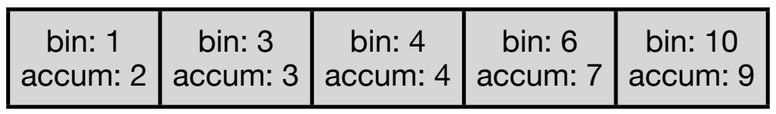
Time Series Encoding



query/tseries/1/3/4

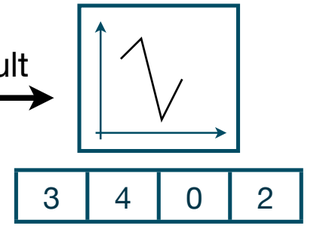
start at bin 1, use buckets of 3 bins each, and collect 4 of these buckets

solve using...



A Summed Table Sparse Representation for Counts

result

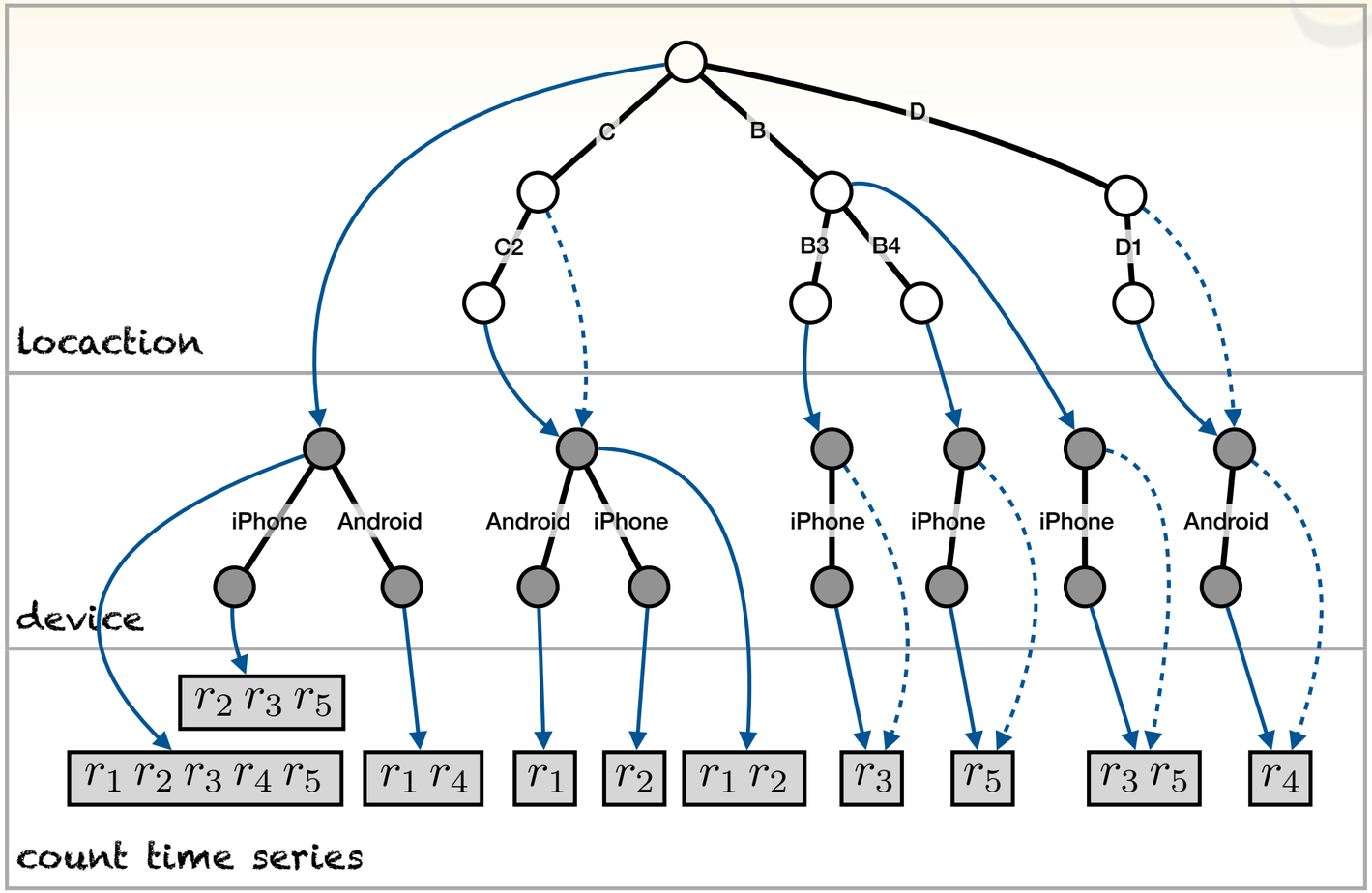


Querying a Nanocube

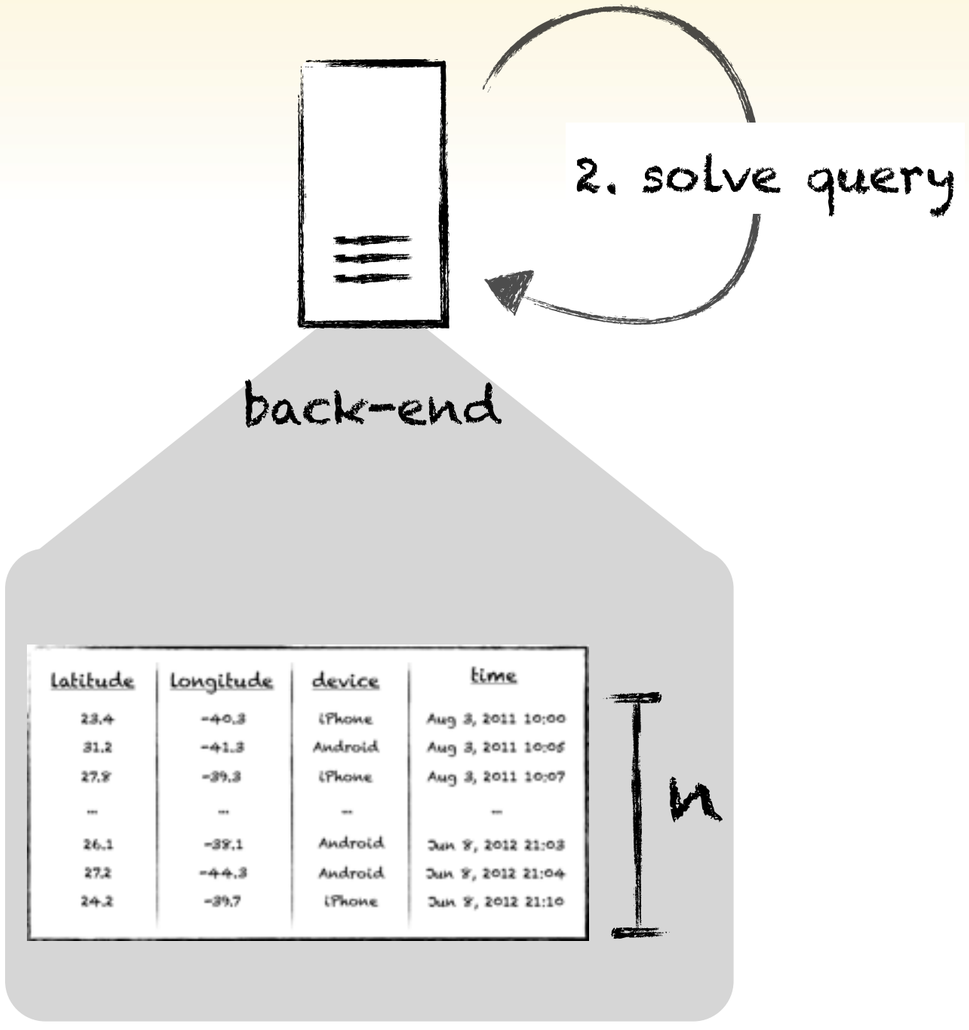
Q1: All Tweets

Q2: All iPhone Tweets

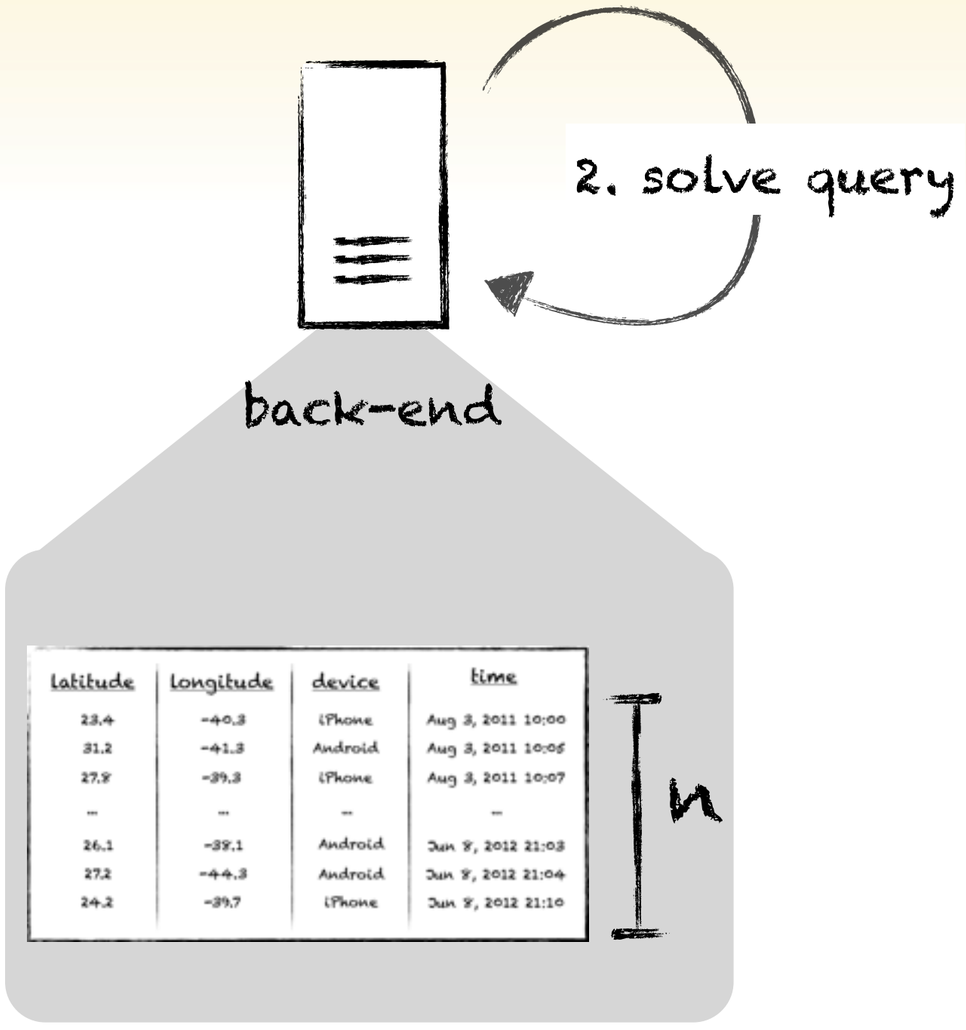
Q2: All Android Tweets in Space Cell C



Solving Queries



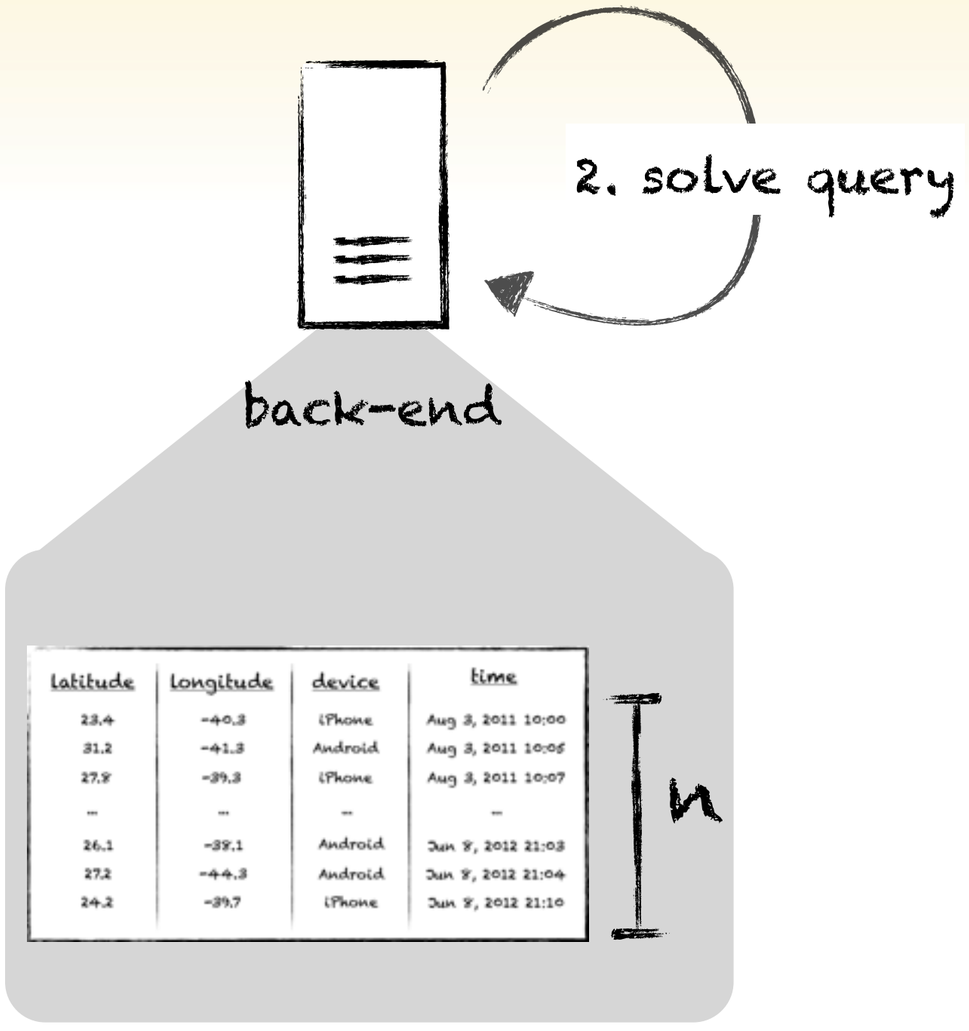
Solving Queries



Approaches

(small) If "n" is small we can scan the data and solve the query

Solving Queries

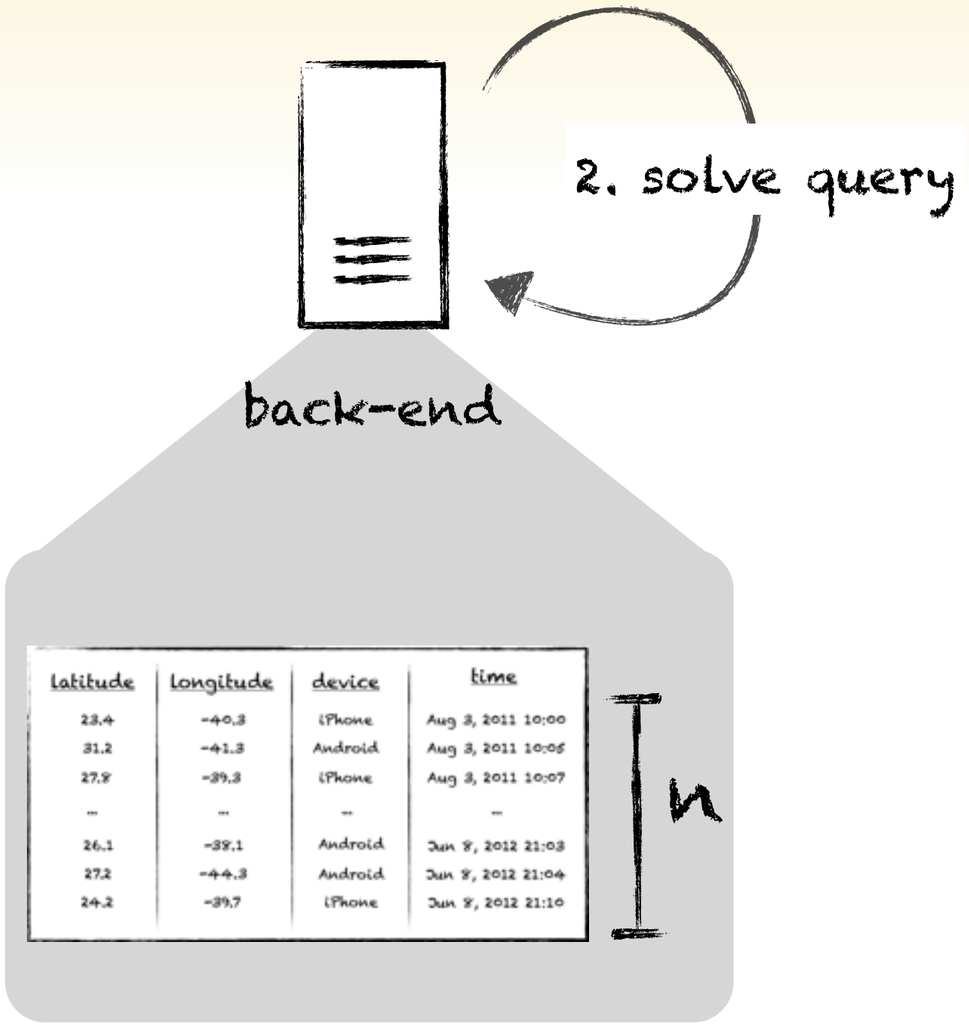


Approaches

(small) If "n" is small we can scan the data and solve the query

(parallel) If "n" is large we can make "k" parallel processors deal with "n/k" data points

Solving Queries



Approaches

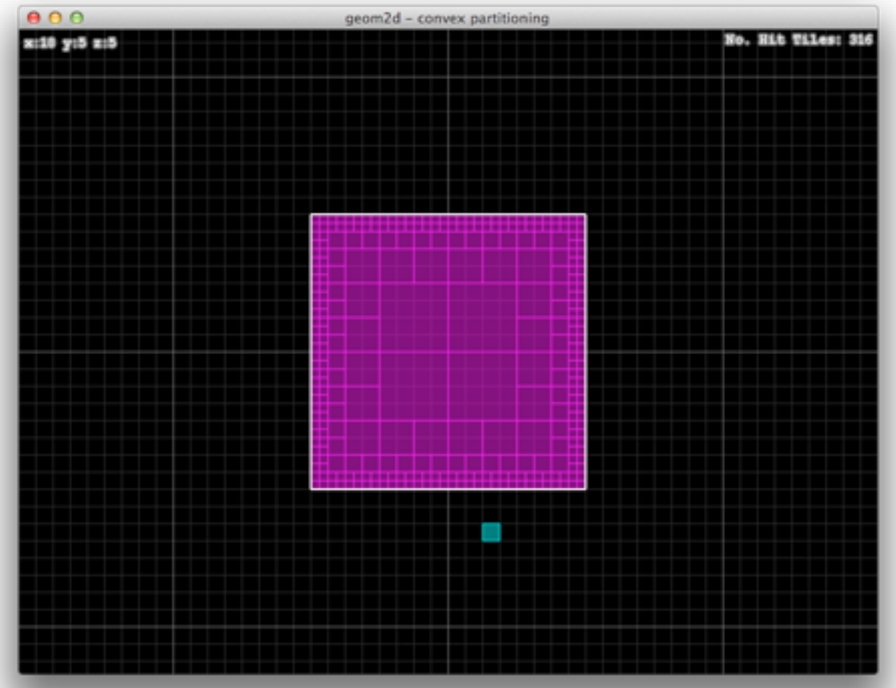
(small) If "n" is small we can scan the data and solve the query

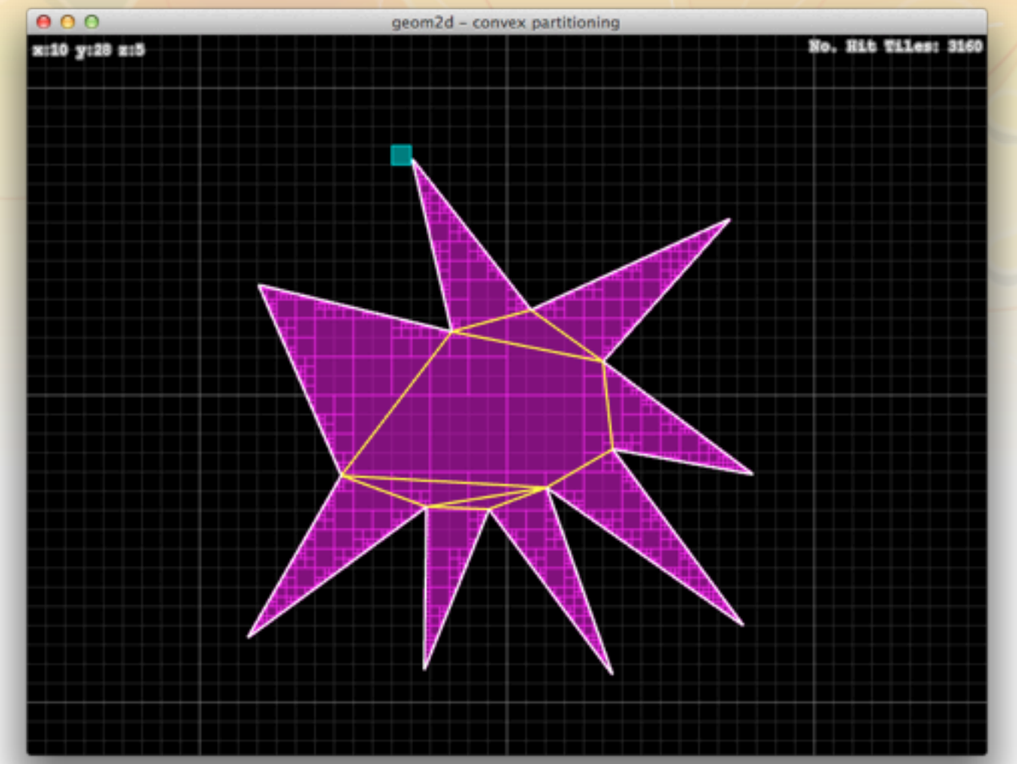
(parallel) If "n" is large we can make "k" parallel processors deal with "n/k" data points

(pre-compute) Pre-compute answers so that at query time computation is not proportional to "n" but to something smaller (e.g. output size)

Demo

What if The Shape of the query doesn't match a predefined shape?





Results

4 to 6 dimensions
(space counting 2)

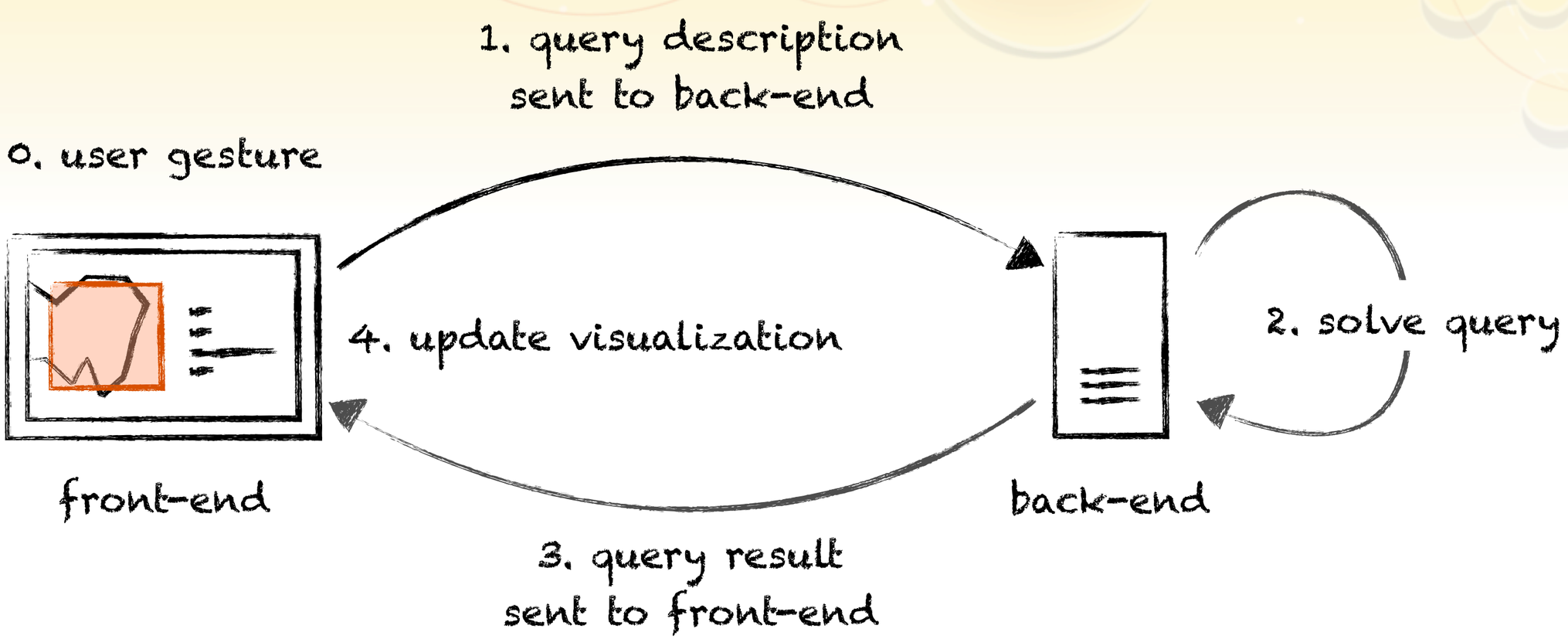
dataset	n	memory	time	keys	cardinality	schema
brightkite	4.5 M	1.6 GB	3.50 m	3.5 M	2^74	spatial(50), time(16), weekday(3), hour(5)
customer tix	7.8 M	2.5 GB	8.47 m	7.8 M	2^69	spatial(50), time(16), type(3)
flights	121 M	2.3 GB	31.13 m	43.3 M	2^75	spatial(50), time(16), carrier(5), delay(4)
twitter-small	210 M	10.2 GB	1.23 h	116.0 M	2^53	spatial(34), time(16), device(3)
twitter	210 M	46.4 GB	5.87 h	136.0 M	2^60	spatial(34), time(16), lang(5), device(3), app(2)
splom-10	1 B	4.3 MB	4.13 h	7.4 K	2^20	d1(4), d2(4), d3(4), d4(4), d5(4)
splom-50	1 B	166 MB	4.72 h	1.9 M	2^30	d1(6), d2(6), d3(6), d4(6), d5(6)
cdrs	1 B	3.6 GB	3.08 h	96.3 M	2^69	spatial(50), time(16), duration(3)

Comparison with imMens

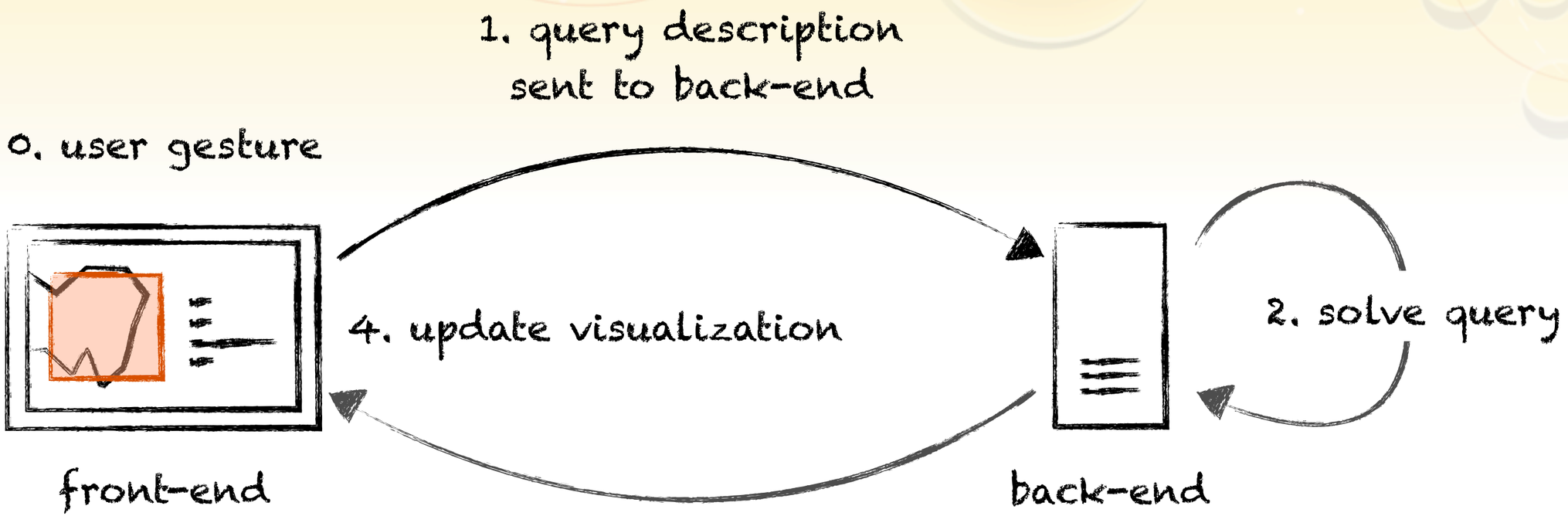
	imMens	Nanocubes
Technique	GPU & GPU Mem.	CPU & Main RAM
Hard Dimension Limit	<= 4 dimensions	no hard limit
Max Bins. Across All Dim	2^30	no hard limit
Frame Rate	60fps	10 to 60fps
Memory Scheme	Dense	Sparse
Supports Streaming	No	Yes

[EuroVis 2013, Z. L. Liu, B. Jiang, and J. Heer.
imMens: Real-time visual querying of big data.]

Interaction Cycle



Interaction Cycle



requirement

cycle time = t1 + t2 + t3 + t4 <= time budget

Interaction Cycle