Towards Building Interactive and Online Analytics Systems

Feifei Li
https://www.cs.utah.edu/~lifeifei/
University of Utah

Interactive and Online Data Analytics Systems

- Interactive query and analytics:
  - *Issue Queries as You Wish*

- Online query and analytics:
  - *Control the tradeoff between result quality and query/analytics efficiency*

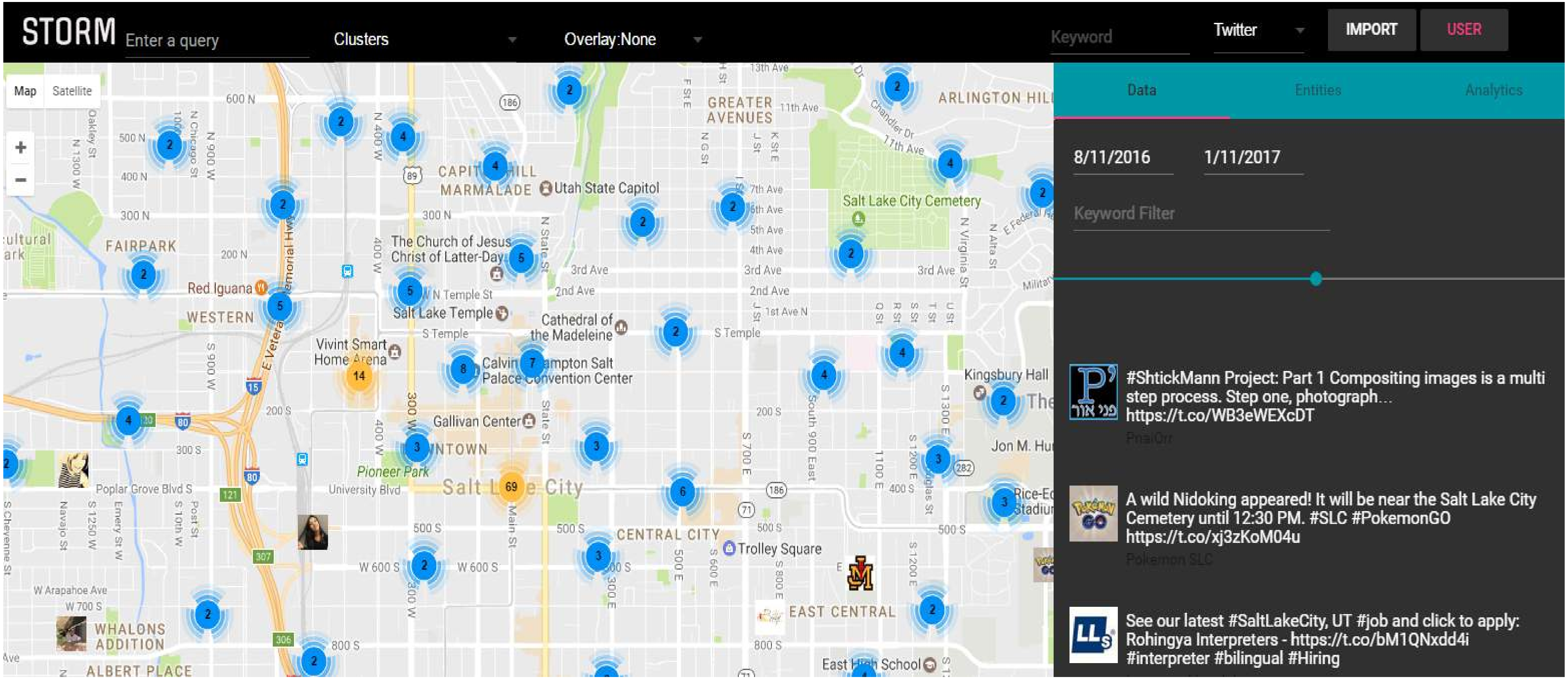
- Rich analytical support:
  - *Support query and analytical operations for knowledge discovery through easy-to-use and intuitive query abstraction and interfaces*

Interactive and Online Data Analytics Systems

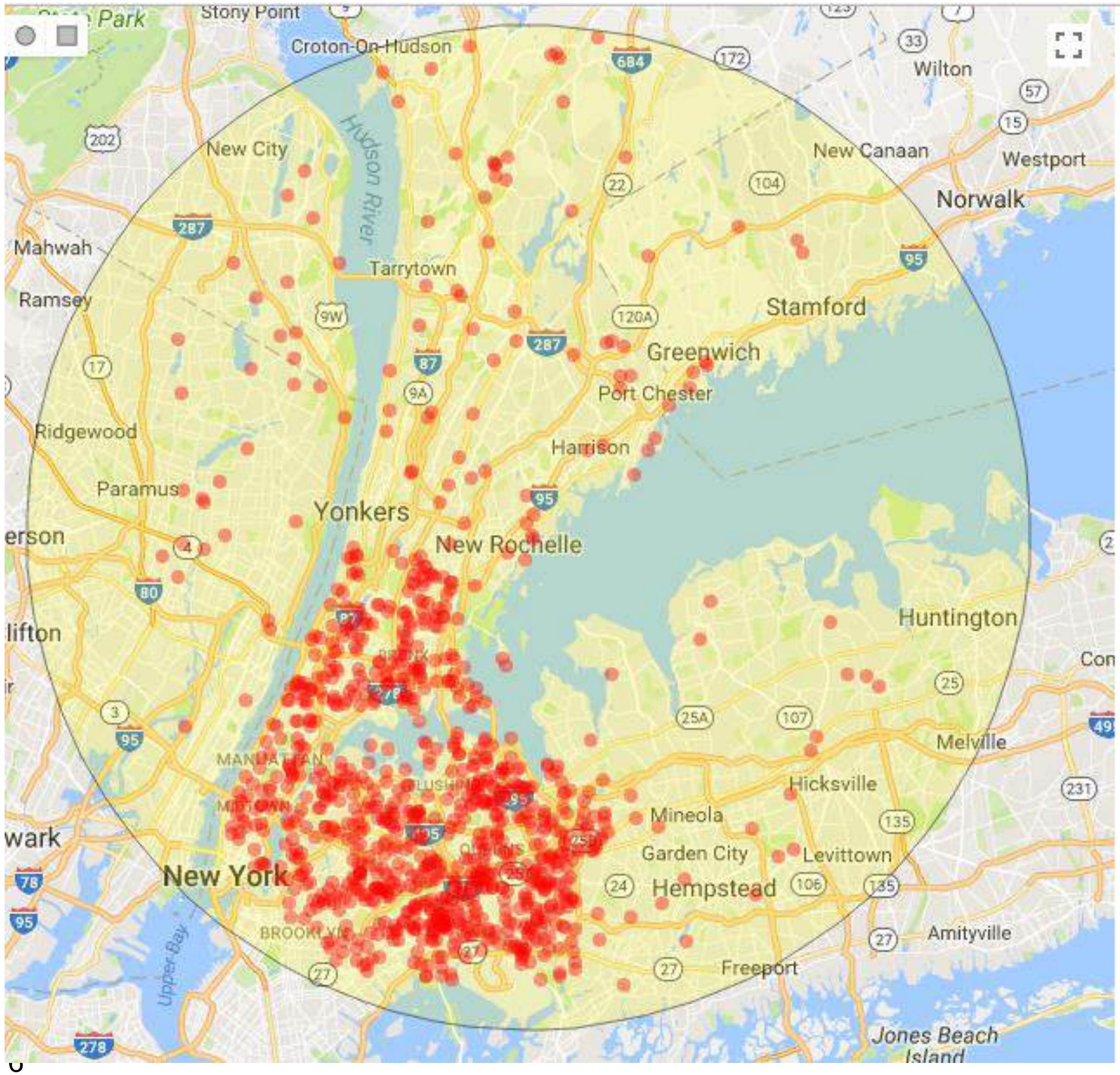
- Geo-tagged tweets as an example (crawling since May 2014)
  - 1.4 geo-tagged billion tweets so far
  - 1.8 TB
  - 3-4 million new tweets per day

- Interactive and online analytics is a must-have:
  - Doing ETL and building a data warehouse is expensive and restrictive

System Interface



Interactive and Online Data Analytics Systems



Query | Options

SQL · DataFrame ☑ Random Samples

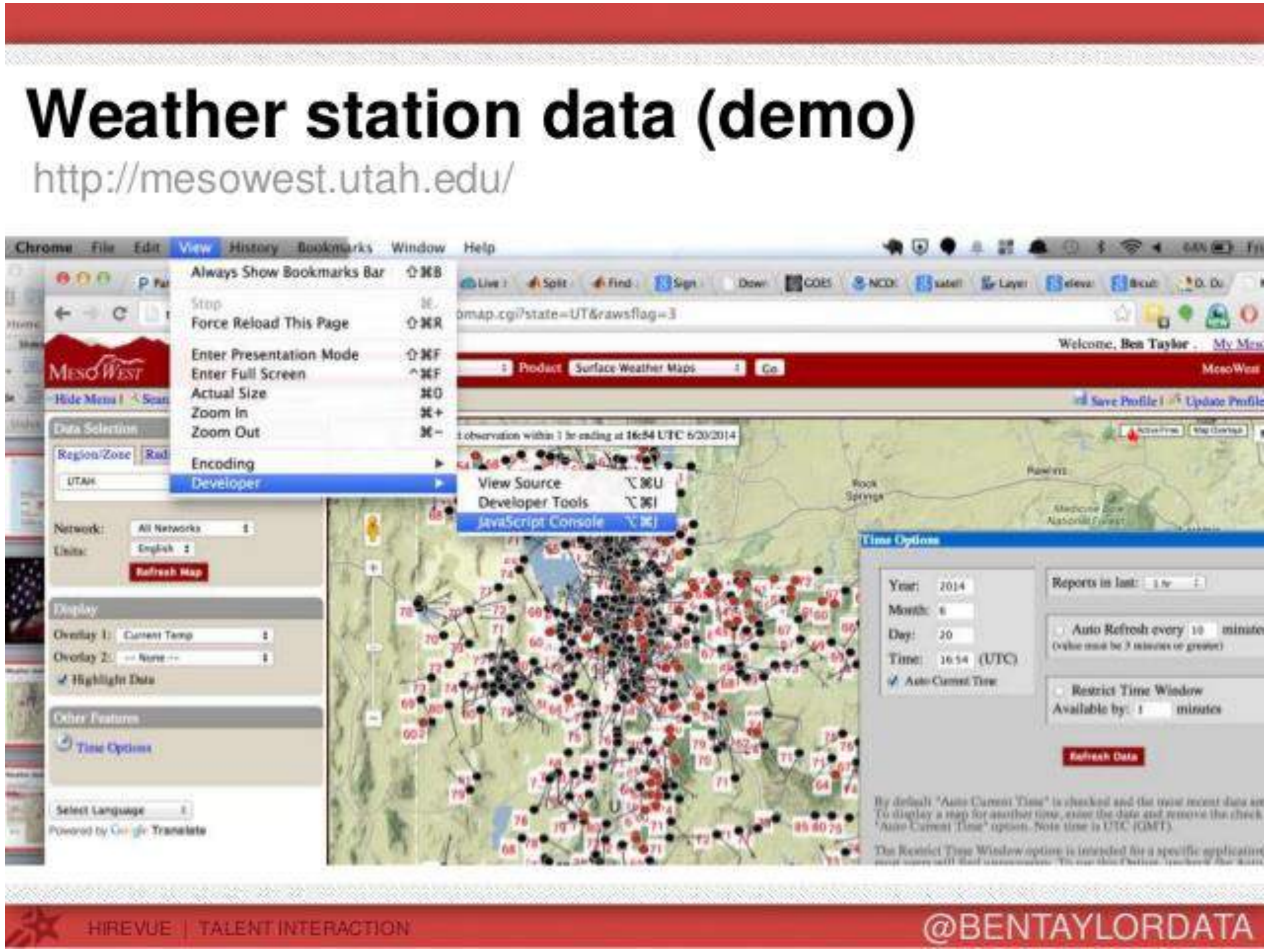
```
SELECT * FROM data
WHERE POINT(lat, long) IN
CIRCLERANGE(POINT(40.92078, -73.76633), 0.30036)
```

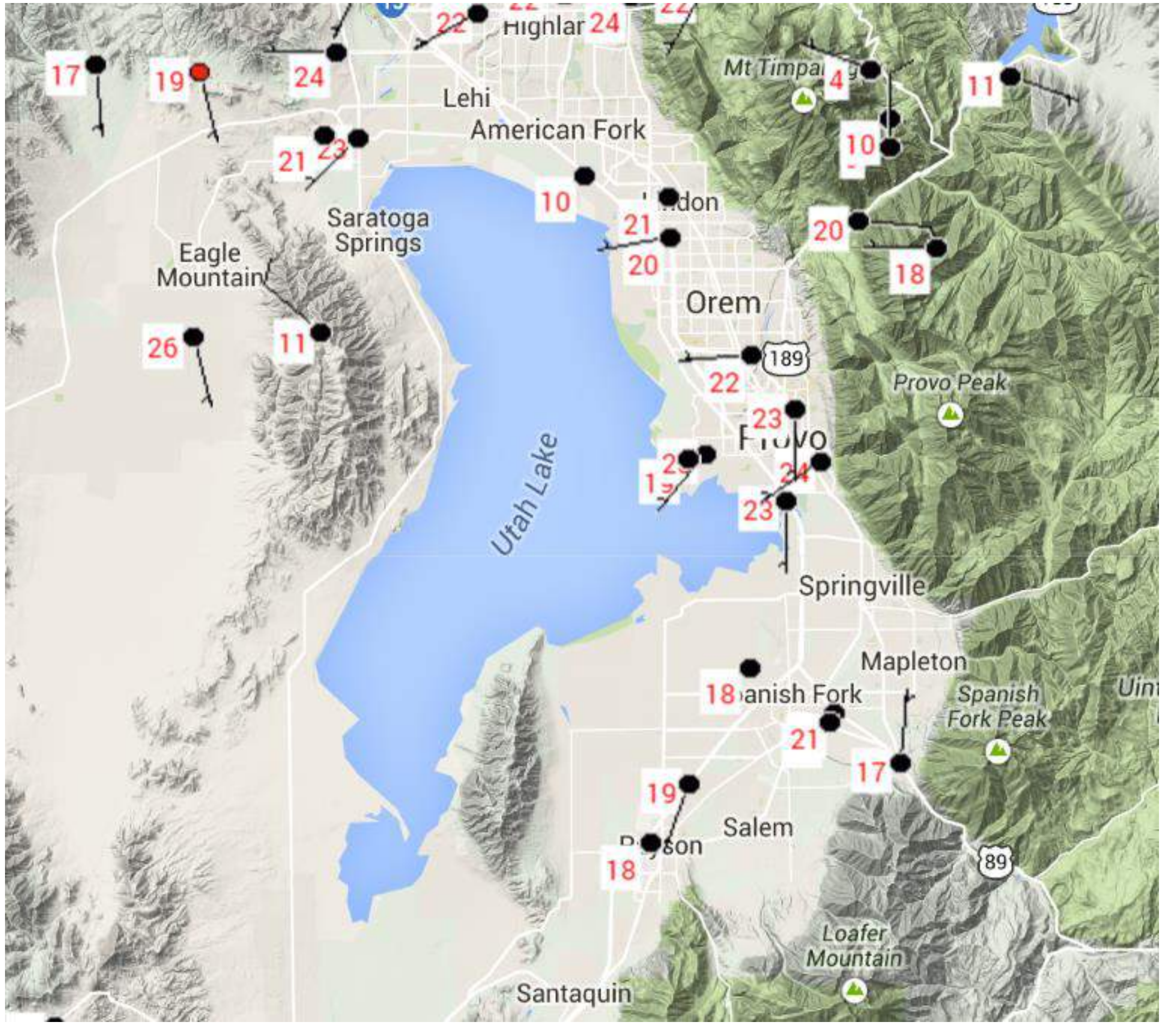
Run Query | Analyze Query

Query Result

id	lat	long	tags
193891132	40.936517	-73.836806	[]
2343908223	40.7641049	-73.825307	[]
2057026674	40.6971534	-73.9071943	[]
193928200	41.029672	-73.690428	[]
2318324437	40.7526274	-73.9459526	[addr:postcode#11101,addr:street#12th Street,addr:housenumber#42-05]

Another example: The MesoWest Project





Interactive and Online Data Analytics Systems

- Key challenges and opportunities
  - **Interactive:** In-Memory Cluster Based Computation
  - **Online:** Accuracy vs. Efficiency Tradeoff: existing systems are binary, either no results or wait for unknown amount of time
  - **Learning:** Real-time Tracking, Monitoring and Prediction: analyzing incoming data in conjunction with historical data (using machine-learning based, data driven approach)

100 TB on 1000 machines

½ - 1 Hour

1 - 5 Minutes

Hard Disks

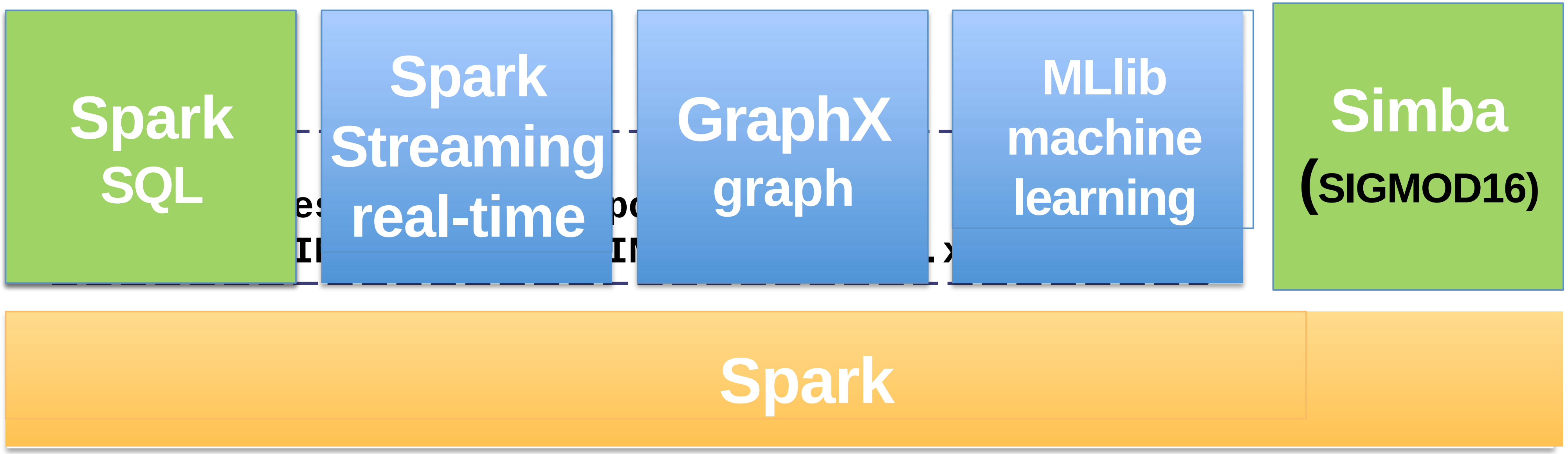
Memory

In memory computation over a cluster

Rich types of queries and analytics: spatial/multimedia data

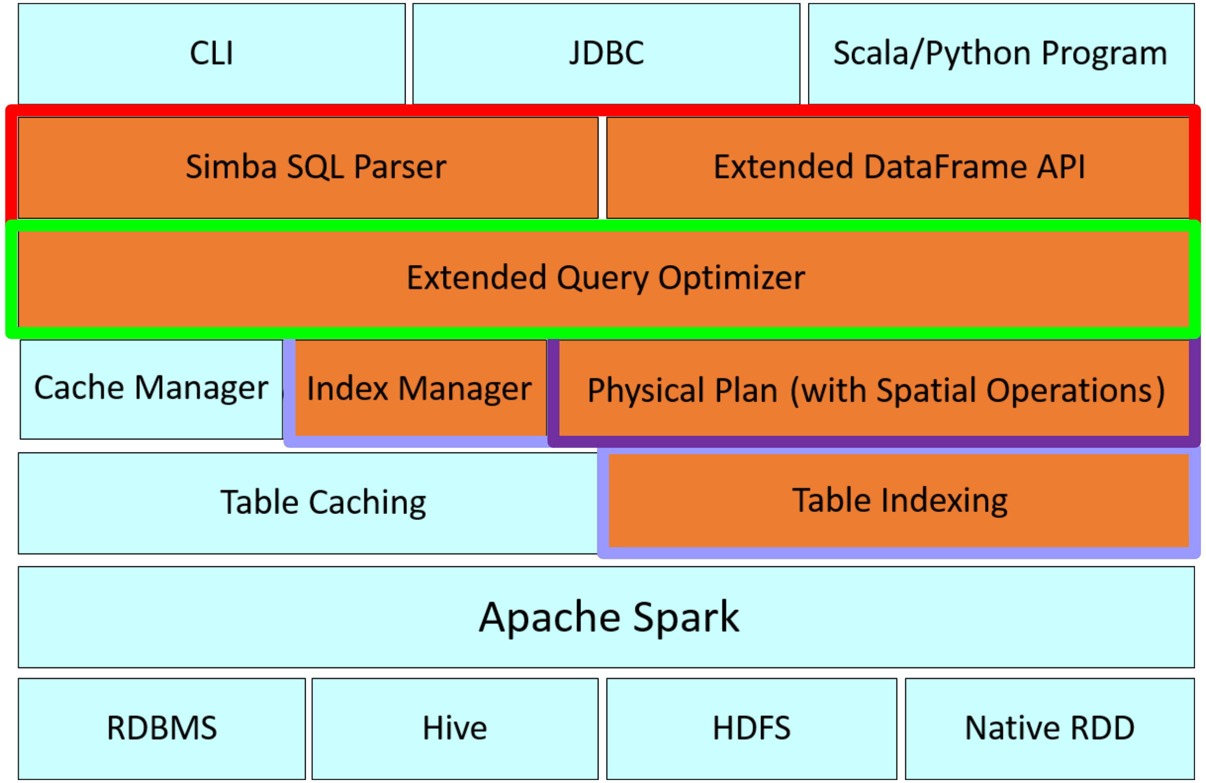
```
SELECT *
FROM points
SORT BY (x - 2)*(x - 2)
      + (y - 3)*(y - 3)
LIMIT 5
```

```
SELECT *
FROM points
WHERE POINT(x, y)
   IN KNN(POINT(2, 3), 5)
```



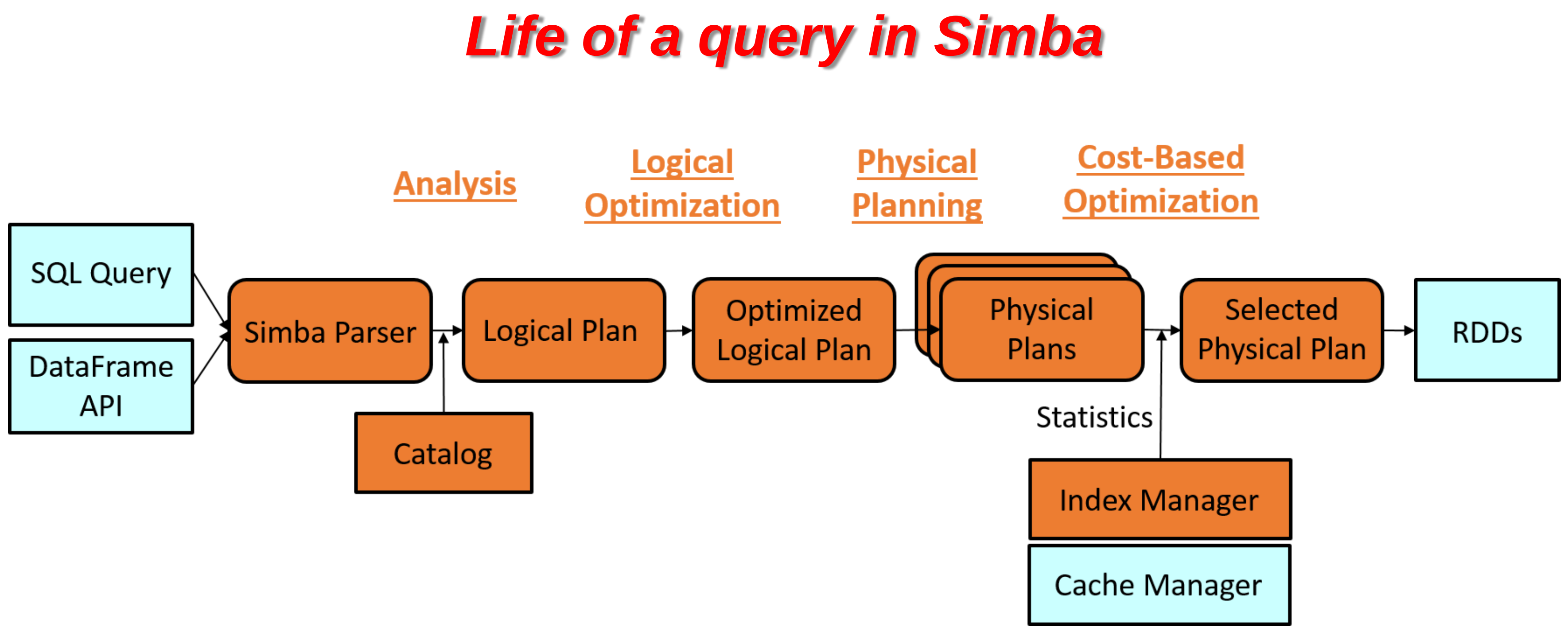
Simba: Spatial In-Memory Big data Analytics

Simba is an extension of Spark SQL across the system stack!



1. *Programming Interface*
2. *Table Indexing*
3. *Efficient Spatial Operators*
4. *New Query Optimizations*

Query Workload in Simba



Programming Interfaces

- Extends both SQL Parser and DataFrame API of Spark SQL
- <u>Make spatial queries more natural</u>

```
SELECT *
FROM points
SORT BY (x - 2)*(x - 2)
      + (y - 3)*(y - 3)
LIMIT 5
```

```
SELECT *
FROM points
WHERE POINT(x, y)
   IN KNN(POINT(2, 3), 5)
```

- <u>Achieve something that is impossible in Spark SQL.</u>

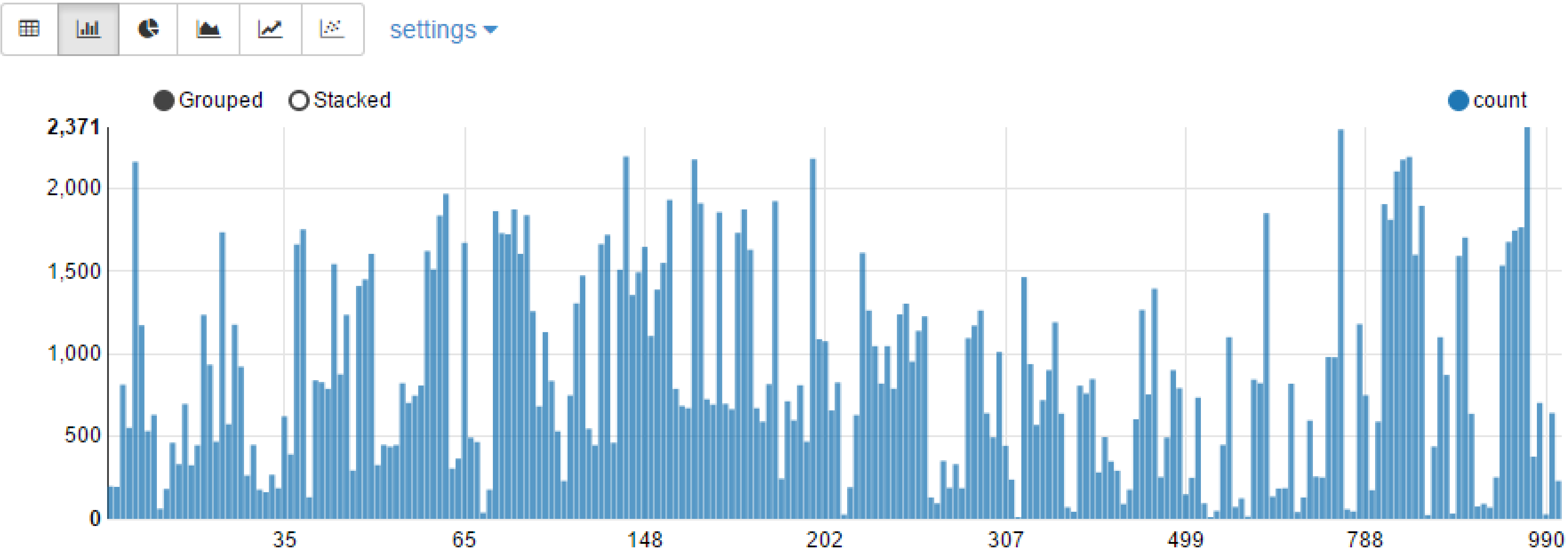
```
SELECT *
FROM queries q KNN JOIN pois p
     ON POINT(p.x, p.y) IN KNN(POINT(q.x, q.y), 3)
```

Zeppelin integration

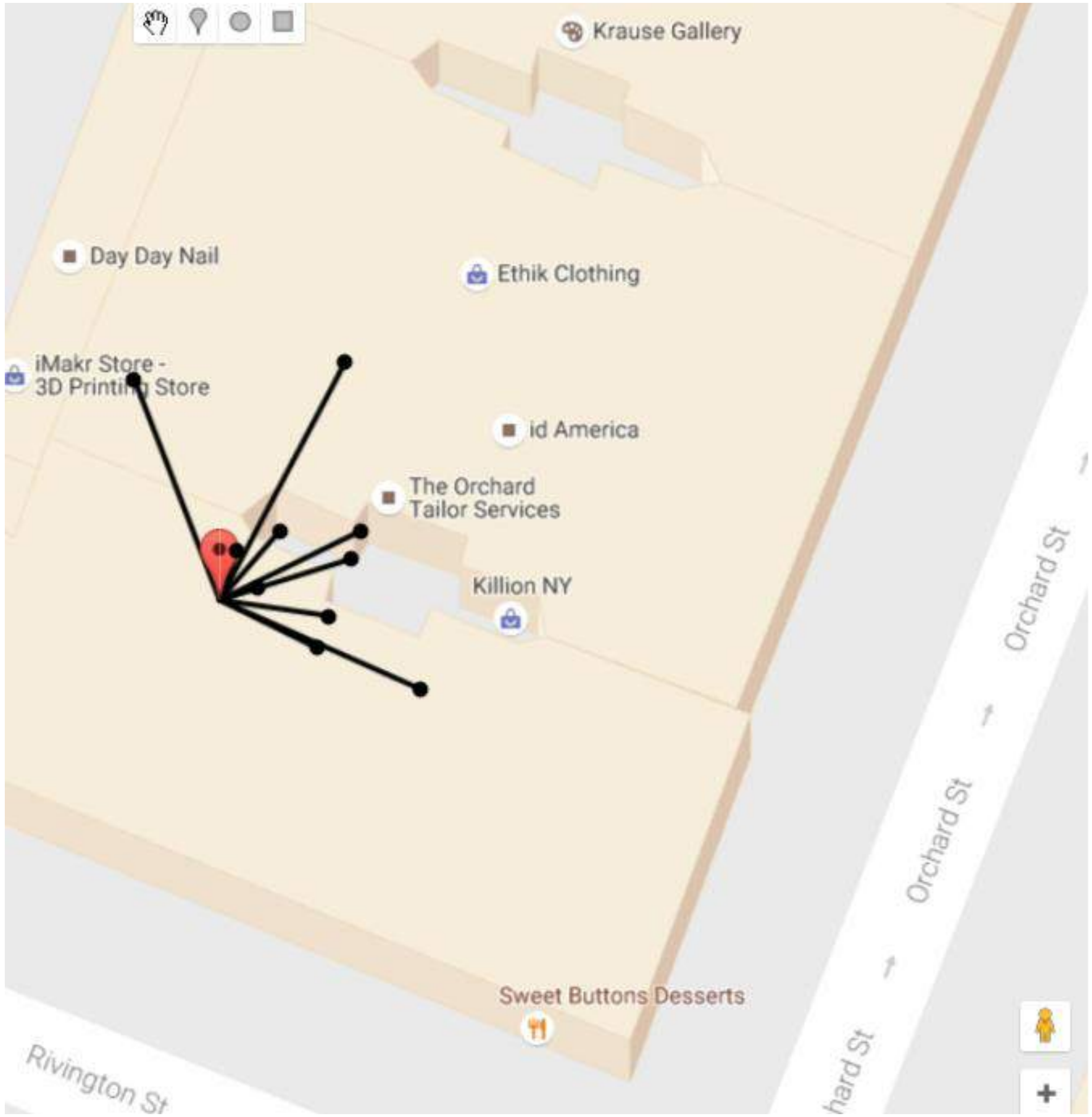
“A web-based notebook that enables interactive data analytics.”

FINISHED

```
%sql
SELECT poi.id, count(*) as count
FROM poi DISTANCE JOIN data
ON POINT(data.lat, data.long) IN CIRCLERANGE(POINT(poi.lat, poi.long), 3)
WHERE POINT(data.lat, data.long) IN RANGE(POINT(24.39, 66.88), POINT(49.38, 124.84))
GROUP BY poi.id
ORDER BY poi.id
```



Query and Analytical Interface



Query | Options

SQL | DataFrame | Random Samples

```sql
SELECT * FROM data
WHERE POINT(lat, long) IN
KNN(POINT(40.72053, -73.98932), 10)
```

Run Query | Analyze Query

Query Result

id	lat	long	tags
2048022149	40.7205381	-73.9892652	[]
2048022152	40.7205461	-73.9892921	[]
2048022150	40.7205404	-73.9893086	[]
2048022163	40.7205945	-73.9892674	[]
2048022139	40.7205009	-73.9892392	[]
2048022145	40.7205218	-73.9892736	[]
2048022047	40.7205893	-73.9893469	[addr:postcode#10002 Street addr:housenuml

Query optimization

Run Query | Analyze Query

Query Result

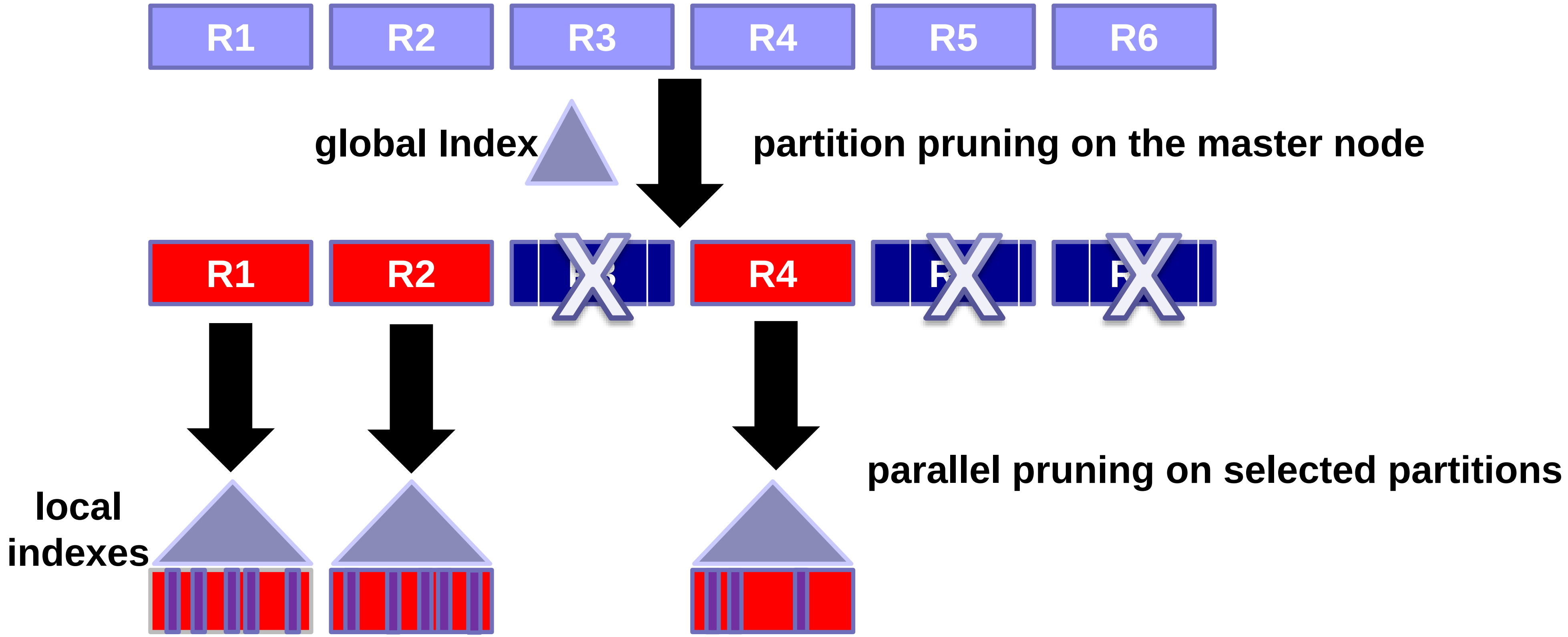
```
== Analyzed Logical Plan ==
id: bigint, lat: double, long: double, tags: string
Project [id#4L,lat#5,long#6,tags#7]
+- Filter  **(pointwrapperexpression(lat#5,long#6)) IN KN
N (POINT(40.72053,-73.98932)) within (10)
   +- Subquery data
      +- LogicalRDD [id#4L,lat#5,long#6,tags#7], MapParti
tionsRDD[11] at rddToDataFrameHolder at <console>:32

== Optimized Logical Plan ==
Filter  **(pointwrapperexpression(lat#5,long#6)) IN KNN
(POINT(40.72053,-73.98932)) within (10)
+- RTreeIndexedRelation [id#4L,lat#5,long#6,tags#7], Scan
ExistingRDD[id#4L,lat#5,long#6,tags#7] , Some(data), [lat
#5,long#6], osm_idx

== Physical Plan ==
IndexedRelationScan [id#4L,lat#5,long#6,tags#7], [ **(poi
ntwrapperexpression(lat#5,long#6)) IN KNN (POINT(40.7205
3,-73.98932)) within (10)], RTreeIndexedRelation [id#4L,l
at#5,long#6,tags#7], Scan ExistingRDD[id#4L,lat#5,long#6,
tags#7] , Some(data), [lat#5,long#6], osm_idx
```

Cost based query optimizations (CBO)

- Indexing support -> efficient algorithms
- Global Index: **partition pruning**
- Local Index: ***parallel*** **pruning within selected partitions**



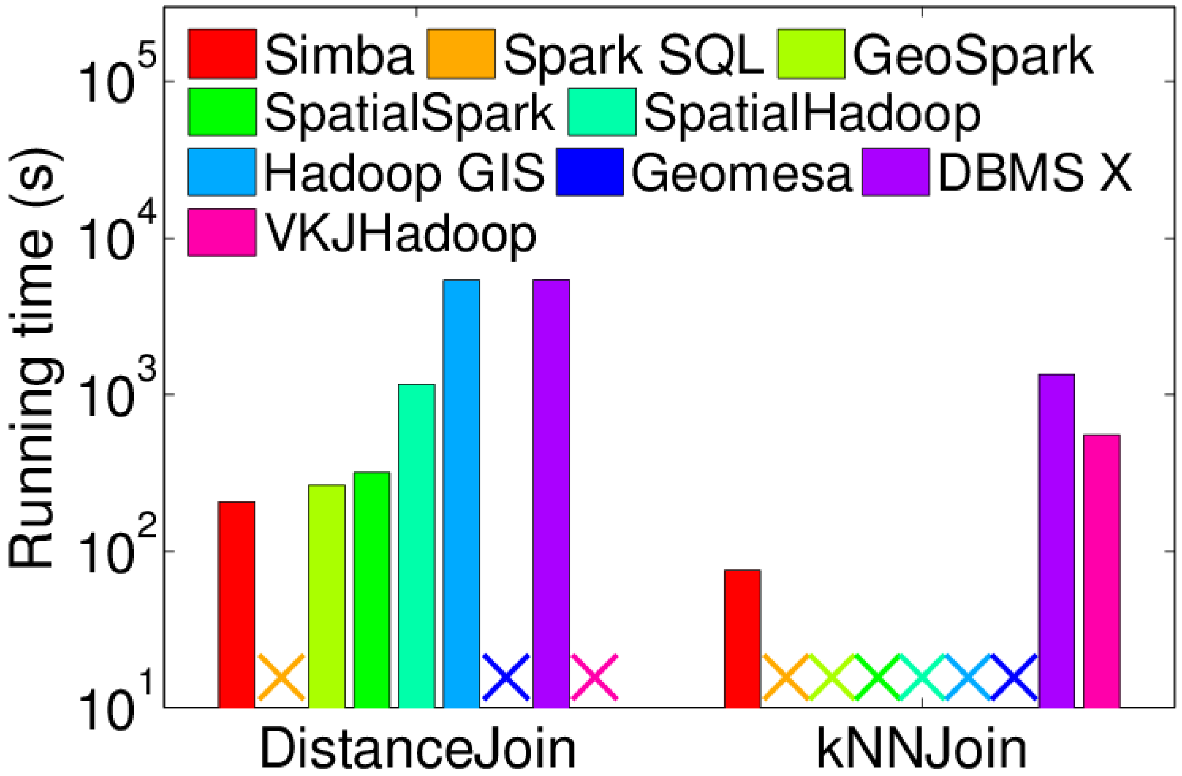
Experiments

- OpenstreetMap Data, 2.7 billion records in 132GB

- 10 nodes with two configurations:
  - 8 machines with a 6-core Intel Xeon E5-2603 v3 1.60GHz processor and 20GB RAM
  - 2 machines with a 6-core Intel Xeon E5-2620 2.00GHz processor and 56GB RAM.

- Other datasets are used in high dimensions
- Open sourced at Github: https://github.com/InitialDLab/Simba
  - currently being used/tested by Hortonworks, Uber, Huawei, ESRI, Alibaba, etc.

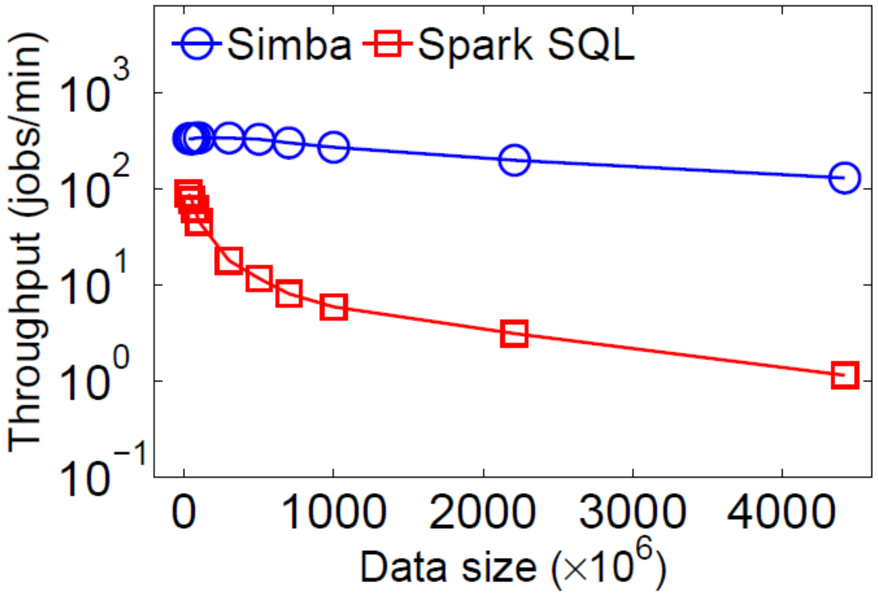
Comparison with Existing Systems (cont'd)



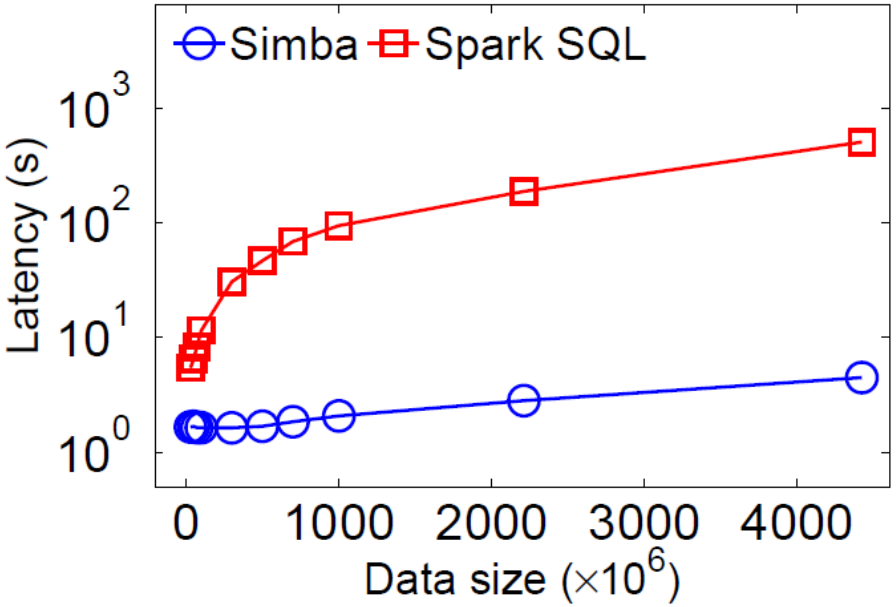
Join operations **performance**

Join between two 3M-entry tables

Performance against Spark SQL: Data Size



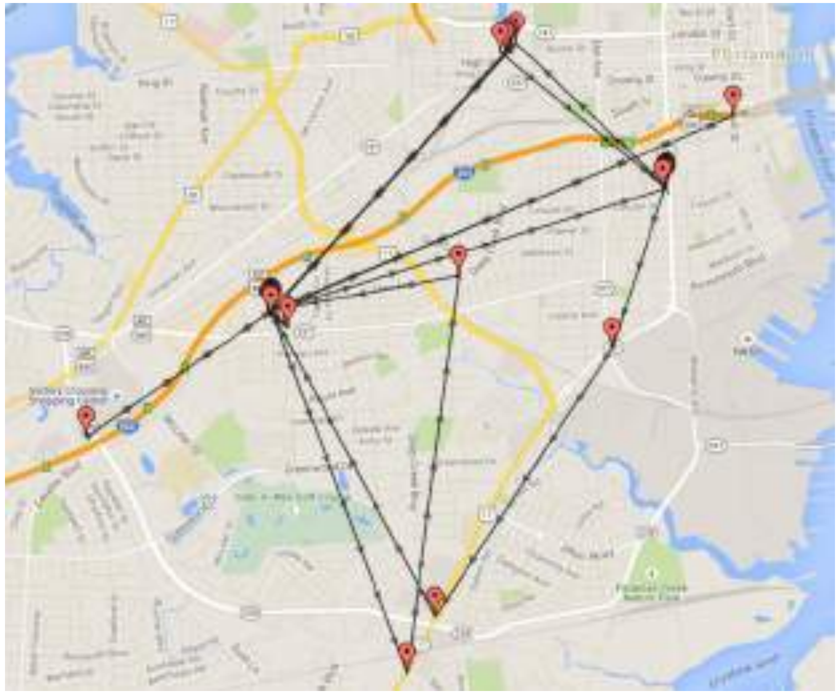
k*NN Query *Throughput



k*NN Query *Latency

Extension: Trajectory Analysis (VLDB 2017)

- Trajectory Data Analysis
  - Massive trajectory retrieval
  - Trajectory Similarity Search



Interactive and Online Data Analytics Systems

- Key challenges and opportunities
  - **Interactive:** In-Memory Cluster Based Computation
  - **Online:** Accuracy vs. Efficiency Tradeoff: existing systems are binary, either no results or wait for unknown amount of time
  - **Learning:** Real-time Tracking, Monitoring and Prediction: analyzing incoming data in conjunction with historical data (using machine-learning based, data driven approach)

Online Analytics

100 TB on 1000 machines

½ - 1 Hour | 1 - 5 Minutes | 1 second

Hard Disks

Memory

?

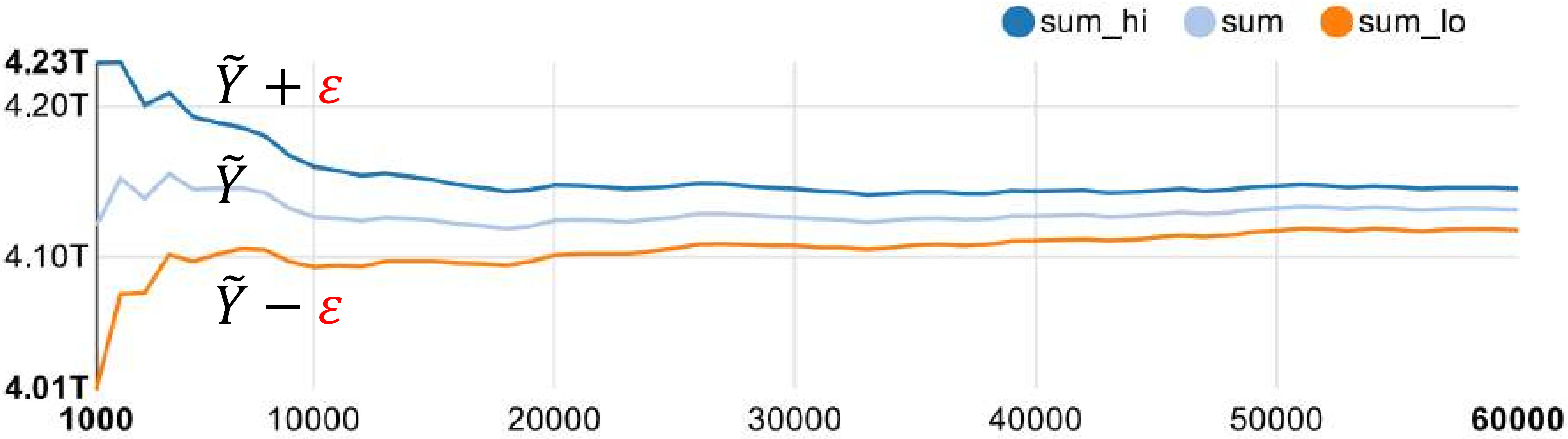
Query Execution on Samples

Complex Analytical Queries (TPC-H)

```
SELECT SUM(l_extendedprice * (1 - l_discount))
FROM customer, lineitem, orders, nation, region
WHERE c_custkey = o_custkey
  AND l_orderkey = o_orderkey
  AND l_returnflag = 'R'
  AND c_nationkey = n_nationkey
  AND n_regionkey = r_regionkey
  AND r_name = 'ASIA'
```

This query finds the total revenue loss due to returned orders in a given region.

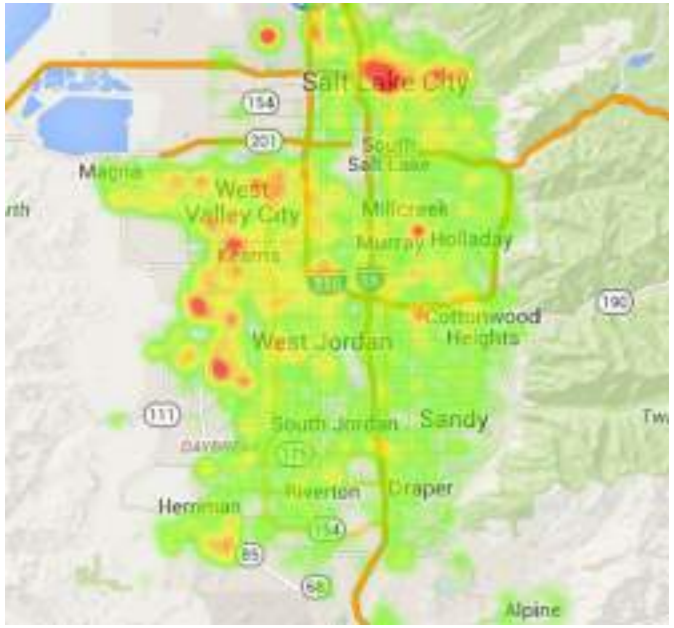
Online Aggregation [Haas, Hellerstein, Wang SIGMOD'97]

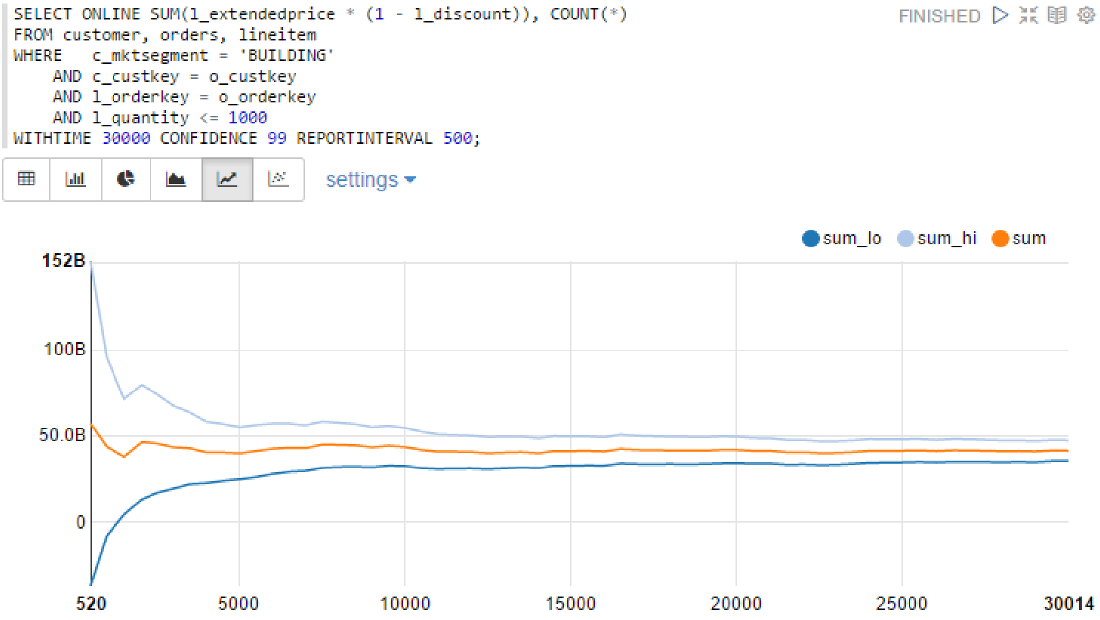


$$\Pr[\underbrace{\tilde{Y} - \varepsilon < Y < \tilde{Y} + \varepsilon}_{\text{Confidence Interval}}] > \underbrace{0.95}_{\text{Confidence Level}}$$

Online spatial and spatio-temporal sampling and analysis (SIGMOD 2015 Best Demo Award, VLDB 2016)

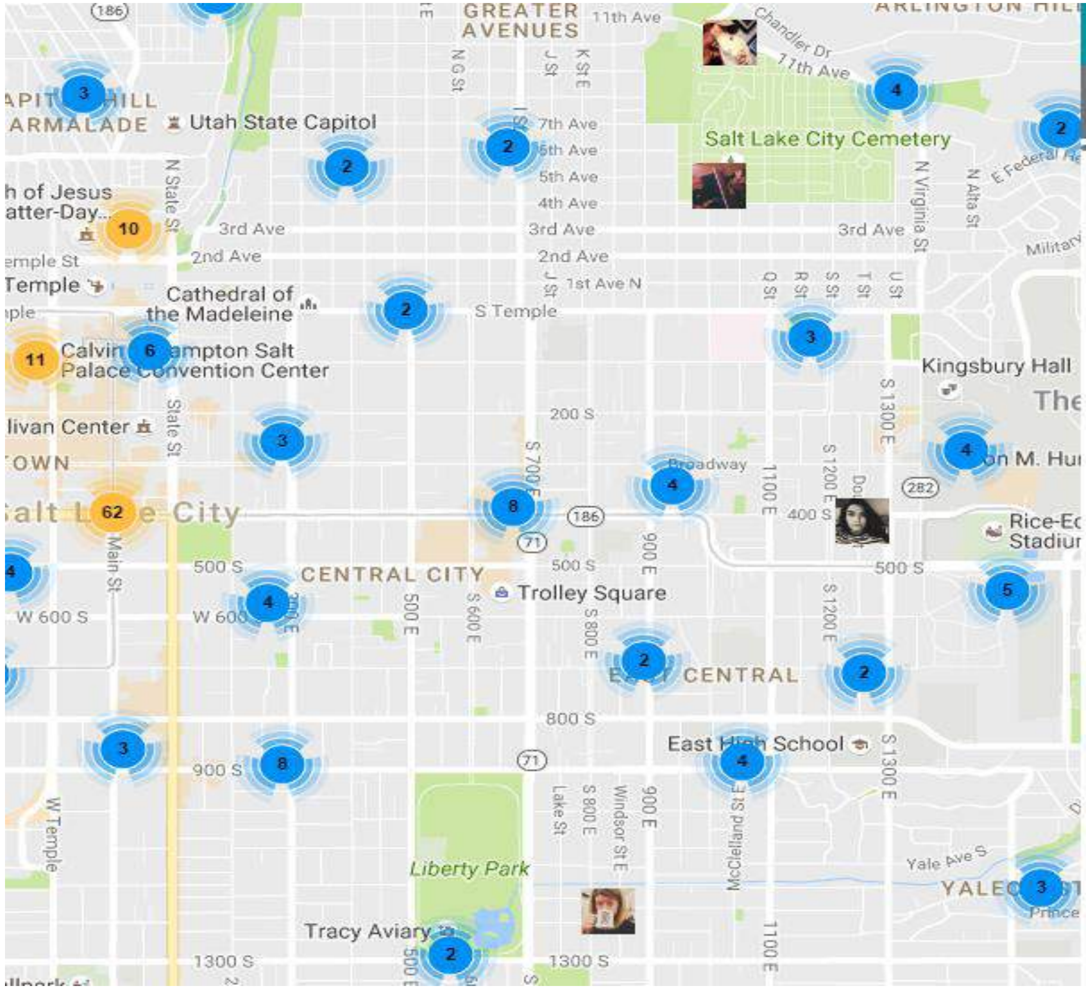
- Online sampling and aggregation support.
  - Integration with the XDB and STORM Projects (both are open sourced on Github)
  - Provides uniform random samples / approximate aggregation results in a online fashion.

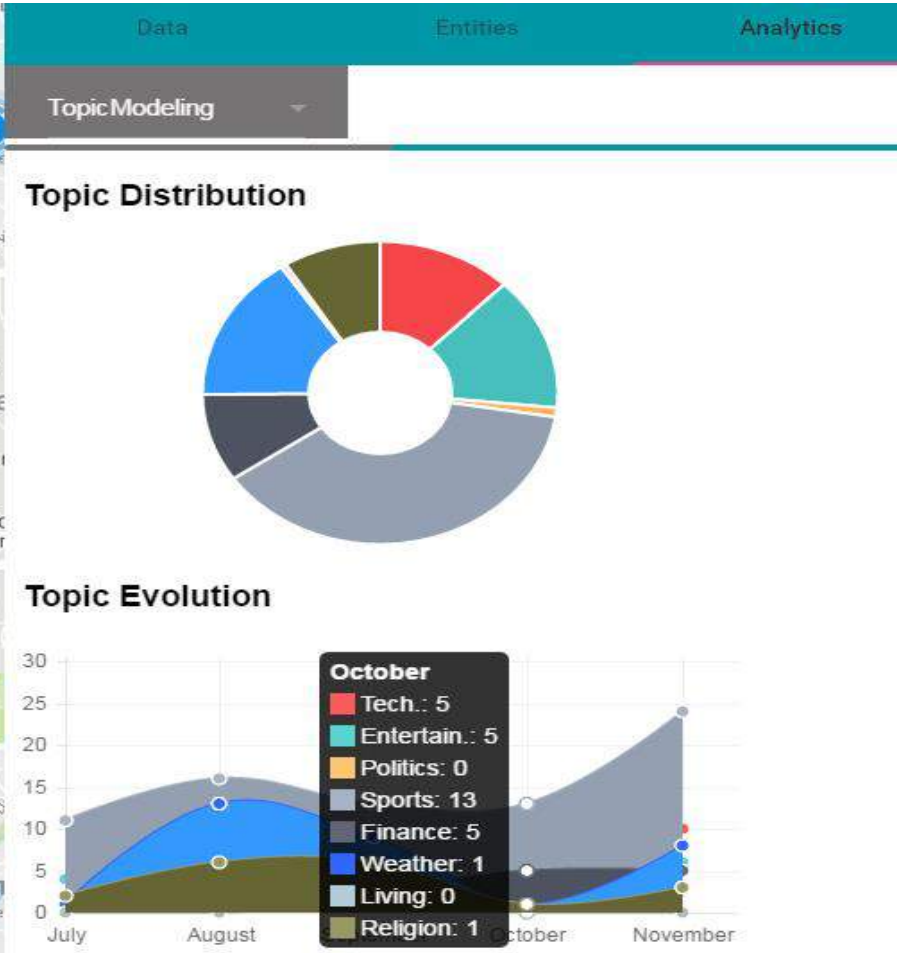




Rich type of online analytics support

- More sophiscated analytics using online samples , e.g., learning, topic modeling, sentiment analysis





Join is even harder

$$\text{sample}(R) \bowtie \text{sample}(S) \neq \text{sample}(R \bowtie S)$$

R
1
2
2
2

$\bowtie$

S
1
1
1
1
2
2

$=$

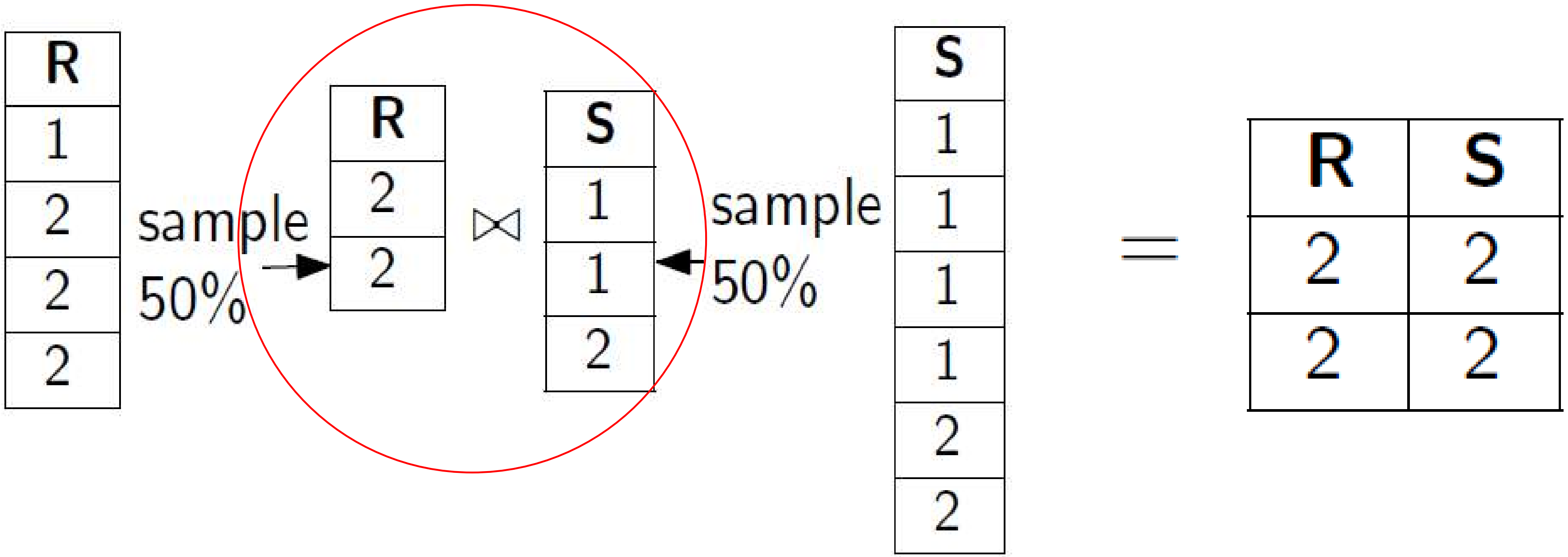
R	S
1	1
1	1
1	1
1	1
2	2
2	2
2	2
2	2
2	2
2	2

50% sample →

R	S
1	1
1	1
2	2
2	2
2	2

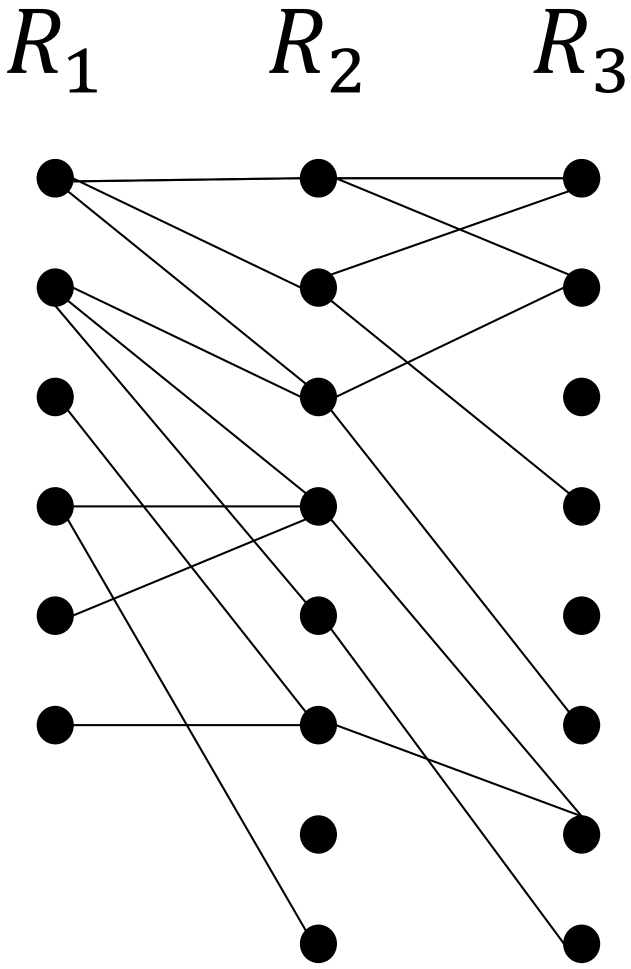
Join is even harder

$$\text{sample}(R) \bowtie \text{sample}(S) \neq \text{sample}(R \bowtie S)$$



Join as a Graph

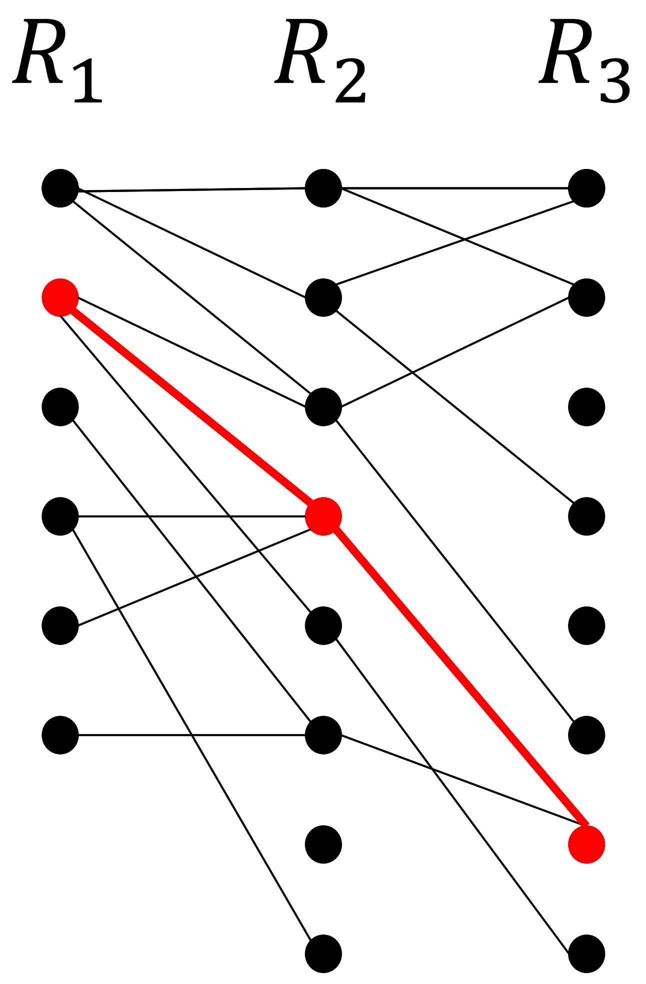
Conceptual only
Never materialized



Join as a Graph – Random Walks: Wander Join (SIGMOD 2016 Best Paper Award)

Conceptual only
Never materialized

Perform Random Walks
over this graph



Join as a Graph

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
[illegible]	[illegible]
C[illegible]	[illegible]
[illegible]	[illegible]
C[illegible]	[illegible]
US	6
China	7
UK	[illegible]
Japa[illegible]	[illegible]
UK	[illegible]

CID	OrderID
4	1
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
5	6
3	7
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

OrderID	ItemID	Price
4	301	$2100
[illegible]	[illegible]	[illegible]00
[illegible]	[illegible]	[illegible]00
[illegible]	[illegible]	[illegible]00
[illegible]	[illegible]	[illegible]30
1	101	$800
2	201	$300
[illegible]	[illegible]	$200
[illegible]	[illegible]	$100
[illegible]	[illegible]	$600

N: size of each table size, e.g., 10^9

n: # tuples taken from each table = # random walks

s: # estimators, e.g., 10^3

$$n = s = 10^3$$

Unbiased estimator: $\dfrac{\$500}{\text{sampling prob.}} = \dfrac{\$500}{1/3 \cdot 1/4 \cdot 1/3}$

Sampling by Random Walks: Independent but not uniform!

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
        I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	
Japa	
UK	

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300

The probability of this path being sampled is 1/3*1/3*1/3=1/27 (recall that the sampling probability of the join path in the previous example was 1/36)

Sampling by Random Walks: Failures are possible too!

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

Sampling by Random Walks: Failures are possible too!

```
SELECT SUM(Price)
FROM Customers C,
     Orders O,
     Items I
WHERE
 C.Nation = 'China'
 C.CID = O.CID
 O.OrderID =
       I.OrderID
```

Nation	CID
US	1
US	2
China	3
UK	4
China	5
US	6
China	7
UK	8
Japan	9
UK	10

CID	OrderID
4	1
3	2
1	3
5	4
5	5
5	6
3	7
5	8
3	9
7	10

OrderID	ItemID	Price
4	301	$2100
2	304	$100
3	201	$300
4	306	$500
3	401	$230
1	101	$800
2	201	$300
5	101	$200
4	301	$100
2	201	$600

A quick demo with XDB

- Implemented with the latest version of PG
- Changes made to parser, query optimizer, and query evaluator
- TPC-H benchmark with roughly 100GB of data.
- Ongoing work of extending this to Spark SQL and a standalone plugin to other commercial DBs

XDB system (approximate DB)

- Two versions available:
  - Kernel version (based on PostgreSQL)
  - Plug-in version (for PG, Orcal, MySQl, Spark SQL)

```
SELECT ONLINE SUM(l_extendedprice * (1 - l_discount))
FROM customer, lineitem, orders, nation
WHERE    c_custkey = o_custkey
     AND l_orderkey = o_orderkey
     AND l_returnflag = 'R'
WITHTIME 60000 CONFIDENCE 95 REPORTINTERVAL 1000;
```

XDB system (approximate DB)

time (ms)	nsamples	nrejected	sum	rel. CI	count	rel. CI
1000	21393	10916	3569910642150.2490	0.019186	97907946.705871429014	0.016313
2000	44416	22444	3586730855773.8836	0.013261	98730673.166257852229	0.011308
3000	71109	35686	3600590499011.0183	0.010486	98864069.377555128985	0.008938
4000	98090	49172	3592717392203.4044	0.008927	98818827.069793972647	0.007612
5000	125142	62772	3600391679526.0518	0.007896	98971635.208020690316	0.006736
6000	152332	76343	3602864111868.8642	0.007154	99162996.448094457199	0.006105
7000	179516	89998	3601934534121.5514	0.006591	99110142.856252365369	0.005624
8000	206594	103810	3598575888627.0625	0.006148	99044487.459001816987	0.005244
9000	233804	117554	3599326643982.6322	0.005780	98993731.891574974812	0.004931
10000	261158	131192	3601211327922.6953	0.005470	99109792.213686759271	0.004665
11000	288386	144890	3597852843520.4237	0.005206	99056834.629677157285	0.004441
12000	315496	158773	3595708335162.0462	0.004976	99035021.044580185507	0.004245
13000	342809	172341	3596327624352.1382	0.004773	99061088.636552460448	0.004072
14000	370149	185909	3596487834132.9988	0.004593	99082039.357131809991	0.003919
15000	397357	199575	3600190559271.9089	0.004432	99132481.122499715210	0.003782
16000	424643	213270	3599863349946.6727	0.004287	99115869.065430552442	0.003658
17000	451881	226987	3599727537973.7470	0.004157	99124452.086438011513	0.003546
18000	479178	240613	3601516118695.4163	0.004037	99122019.790200210895	0.003444
19000	506337	254367	3599757927911.0626	0.003927	99093614.663269813226	0.003350
20000	533554	268163	3599048511739.6974	0.003826	99072461.742934227414	0.003264

VS.

sum	count
3597407507883.3595	99118338

(1 row)

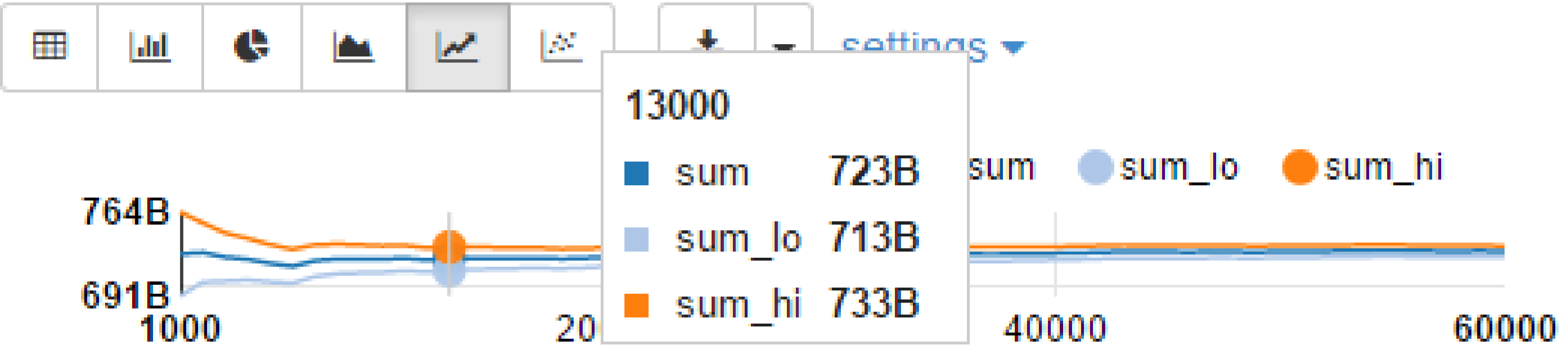
Time: 59874.919 ms

Front-end GUI interface

Revenue loss due to returned goods in a region: wander join

FINISHED

```
%xdb
SELECT ONLINE SUM(l_extendedprice * (1 - l_discount))
FROM customer, lineitem, orders, nation, region
WHERE   c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND l_returnflag = 'R'
    AND c_nationkey = n_nationkey
    AND n_regionkey = r_regionkey
    AND r_name = 'ASIA'
WITHTIME 60000 CONFIDENCE 99 REPORTINTERVAL 1000;
```



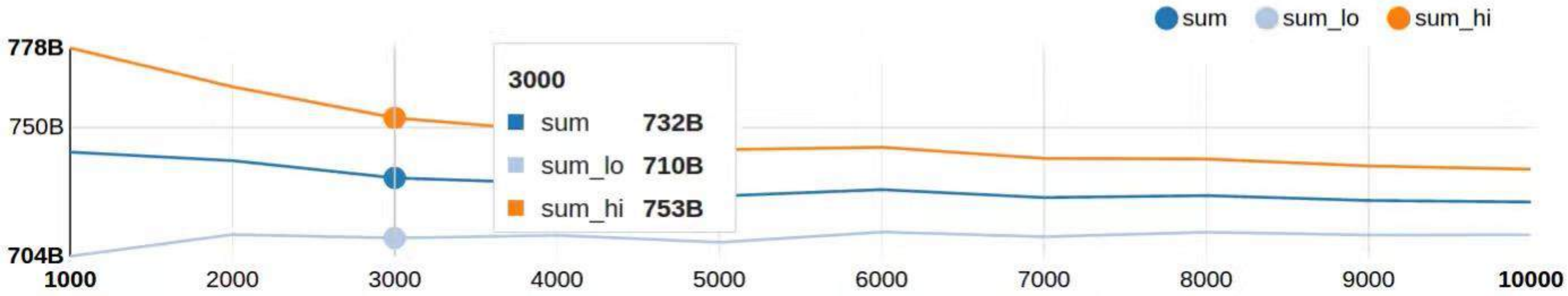
Front-end GUI interface

Revenue loss due to returned goods in a region: wander join

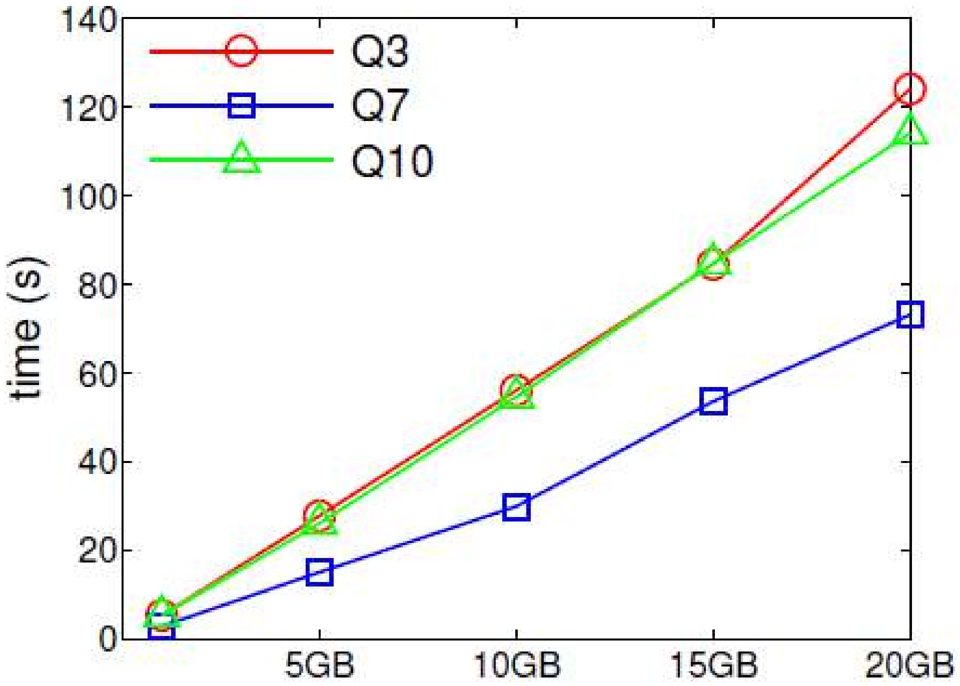
FINISHED

```
%xdb
SELECT ONLINE SUM(l_extendedprice * (1 - l_discount))
FROM customer, lineitem, orders, nation, region
WHERE   c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND l_returnflag = 'R'
    AND c_nationkey = n_nationkey
    AND n_regionkey = r_regionkey
    AND r_name = 'ASIA'
WITHTIME 10000 CONFIDENCE 99 REPORTINTERVAL 1000;
```

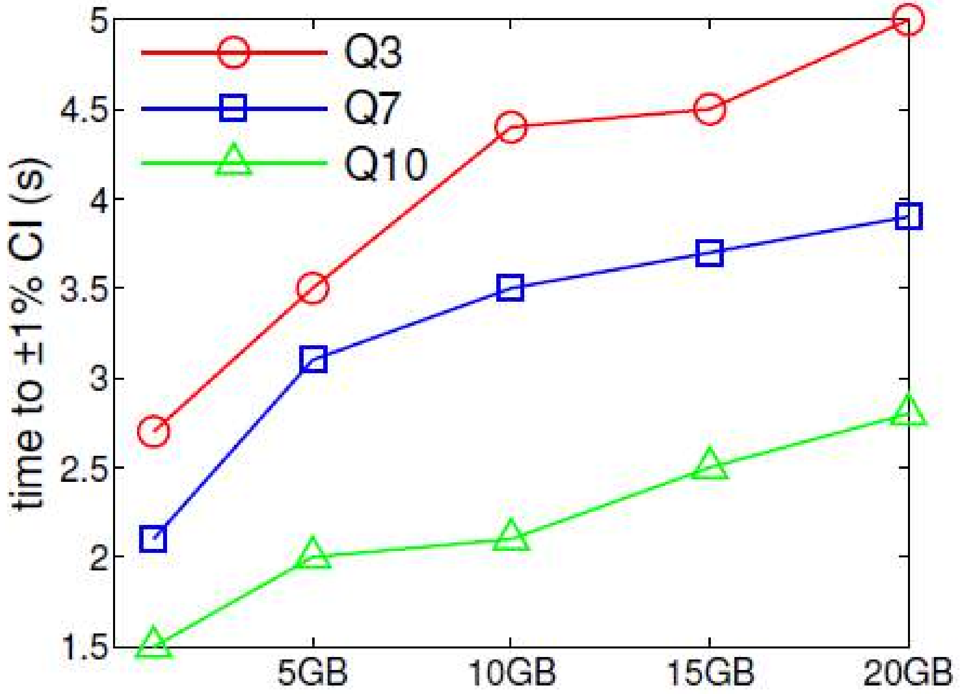
settings



Wander Join in PostgreSQL



(a) PostgreSQL full join



(b) Wander join in PostgreSQL

Logarithmic growth due to B-tree lookup to find random neighbours

Interactive and Online Data Analytics Systems

- Key challenges and opportunities
  - **Interactive:** In-Memory Cluster Based Computation
  - **Online:** Accuracy vs. Efficiency Tradeoff: existing systems are binary, either no results or wait for unknown amount of time
  - **Learning:** Real-time Tracking, Monitoring and Prediction: analyzing incoming data in conjunction with historical data (using machine-learning based, data driven approach)

Beyond aggregations: Integrating Learning Operators

- For Example:

SELECT k-means from Population

WHERE k=8 and feature=age and income >50,000

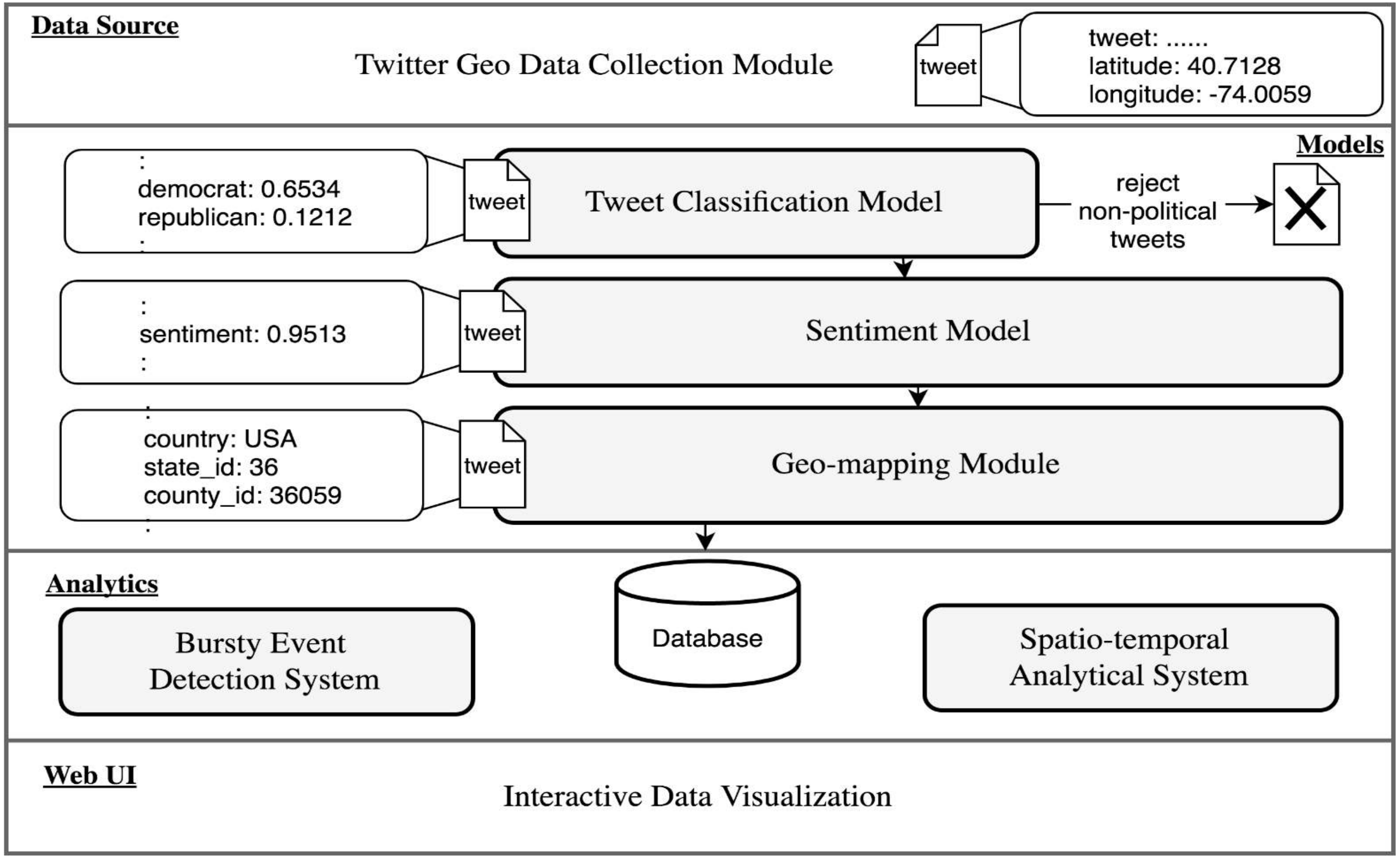
Group By city

What are the impacts to query evaluation and optimization modules?

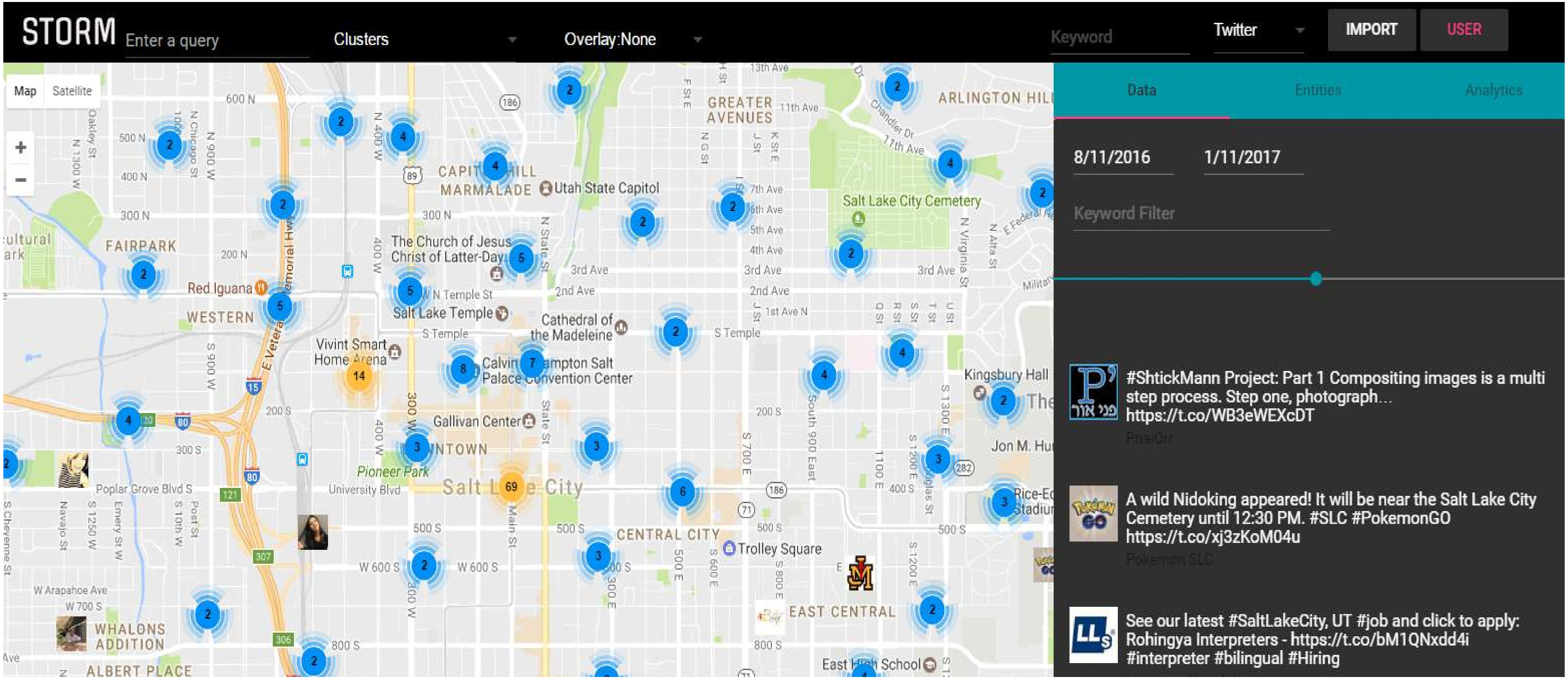
We need a random sample: uniform and independent samples to support "(arbitrary) learning operators over complex queries" (SIGMOD 2018)

Case Study: Large Scale Spatiotemporal Sentiment Analysis (US Election 2016, KDD 2017 Oral Presentation)

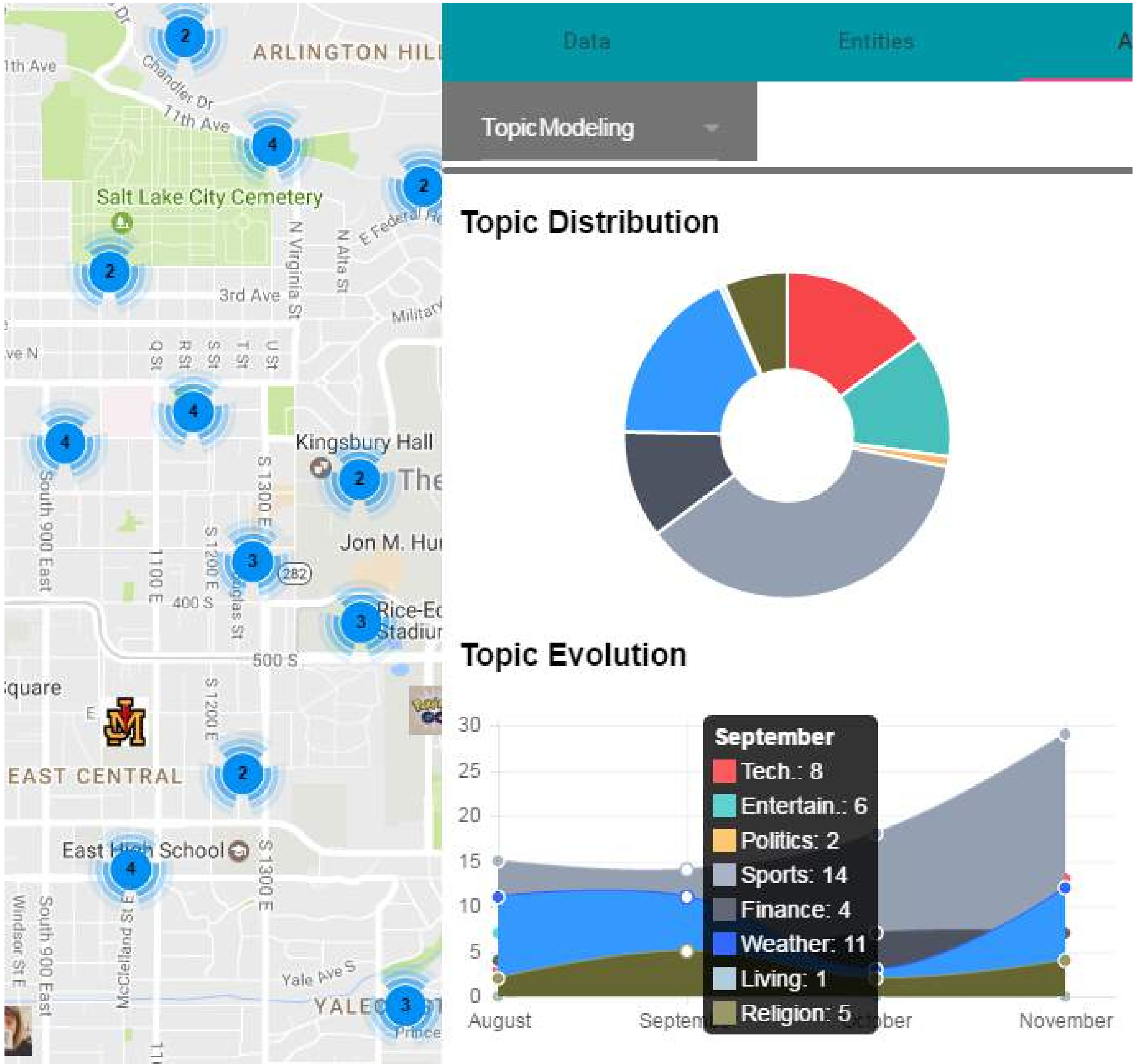
- Compass: System Architecture



System Interface

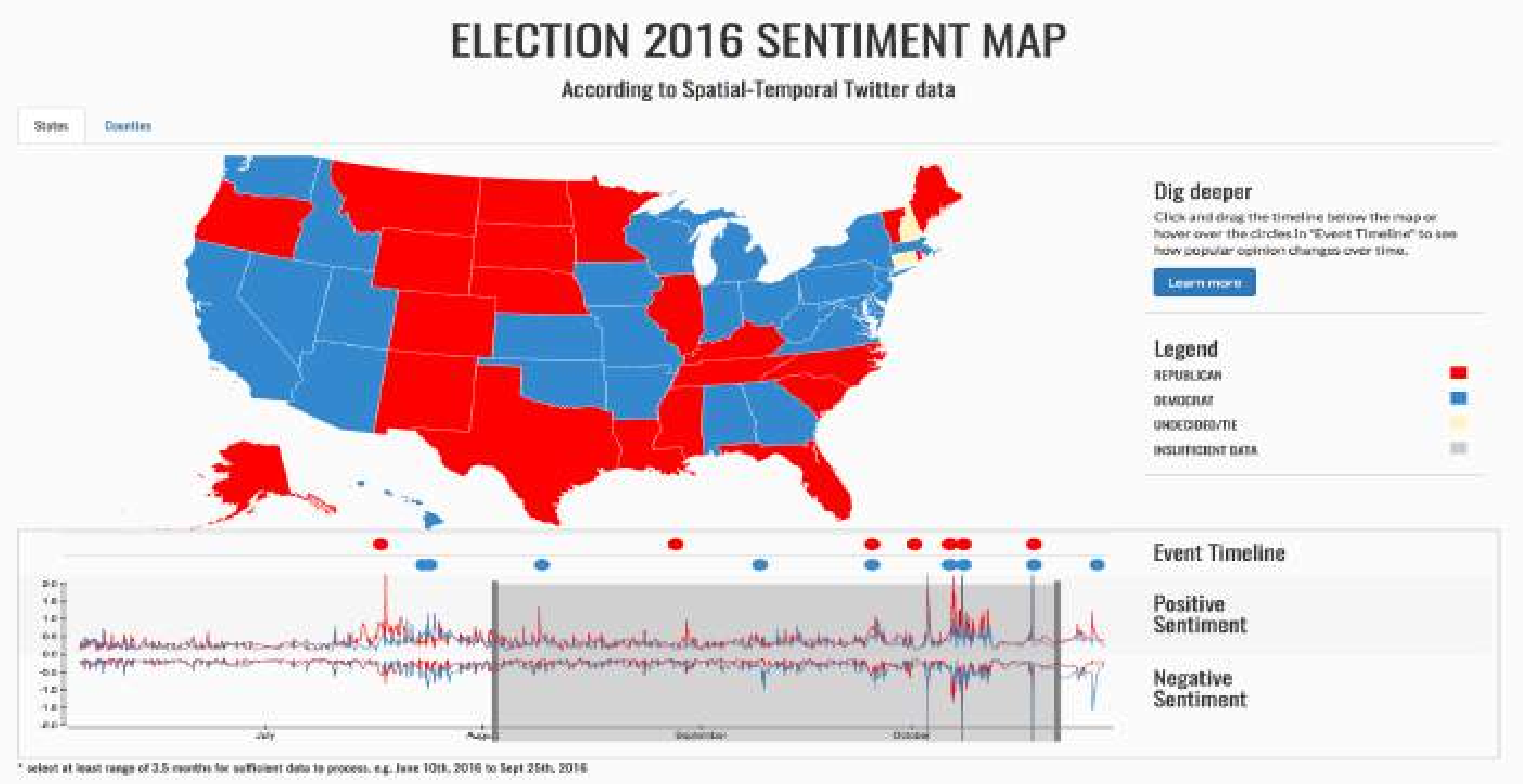


Spatial Temporal Topic Modeling (Google Faculty Award)

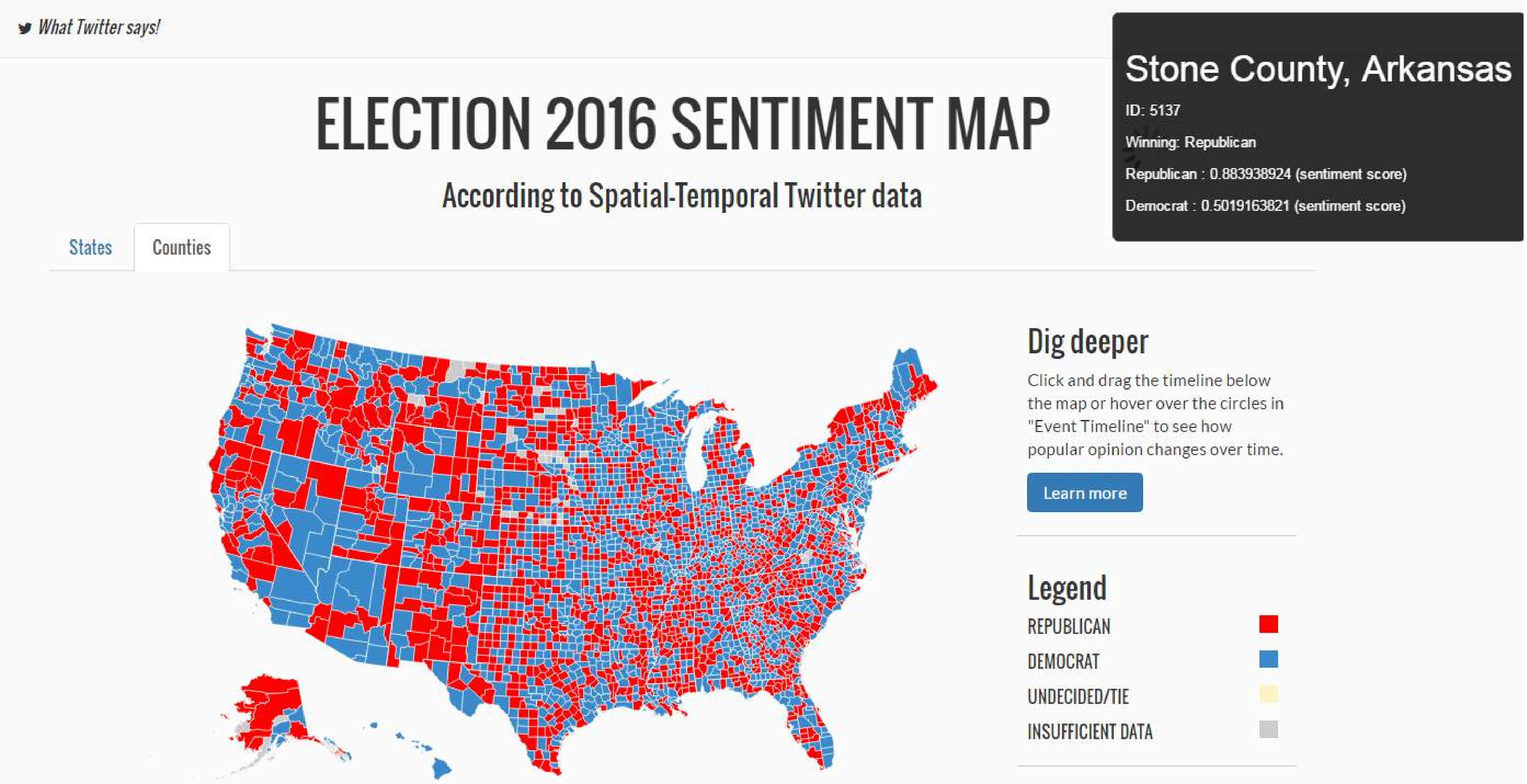


Spatio-temporal Sentiment Analysis (Google Faculty Award)

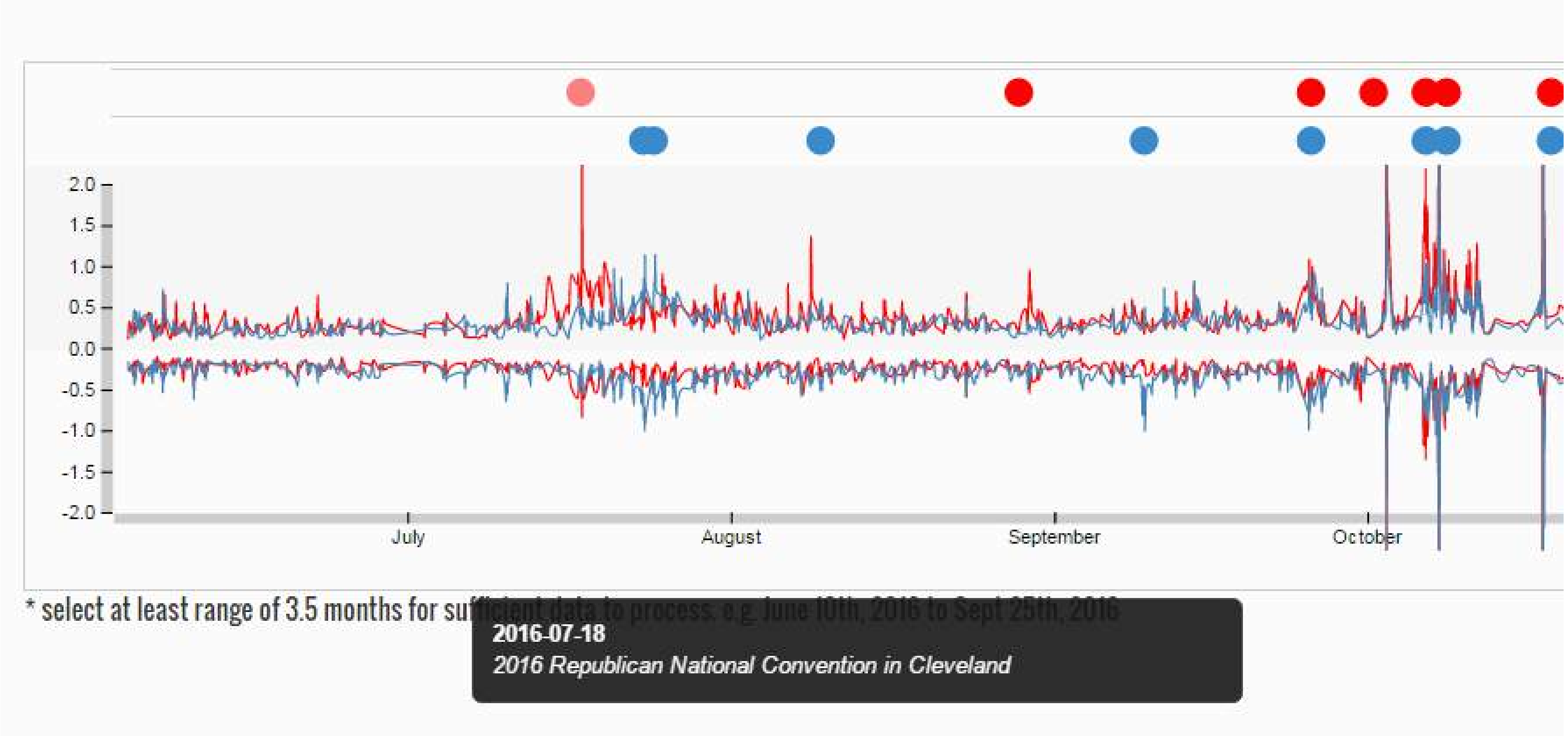
- US Election Sentiment Analysis - http://www.estorm.org



County Level Analysis – Based on Simba

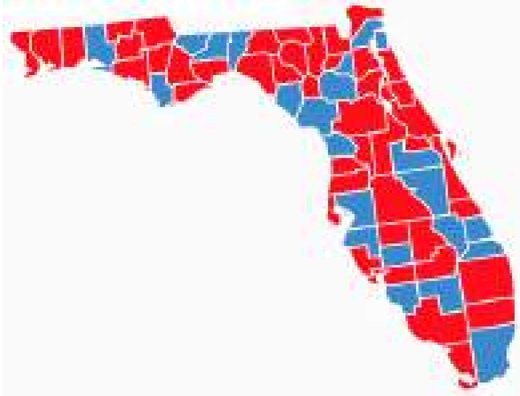


Sentiment Analysis



Geotagged based spatio-temporal sentiment analysis and actual result

Florida

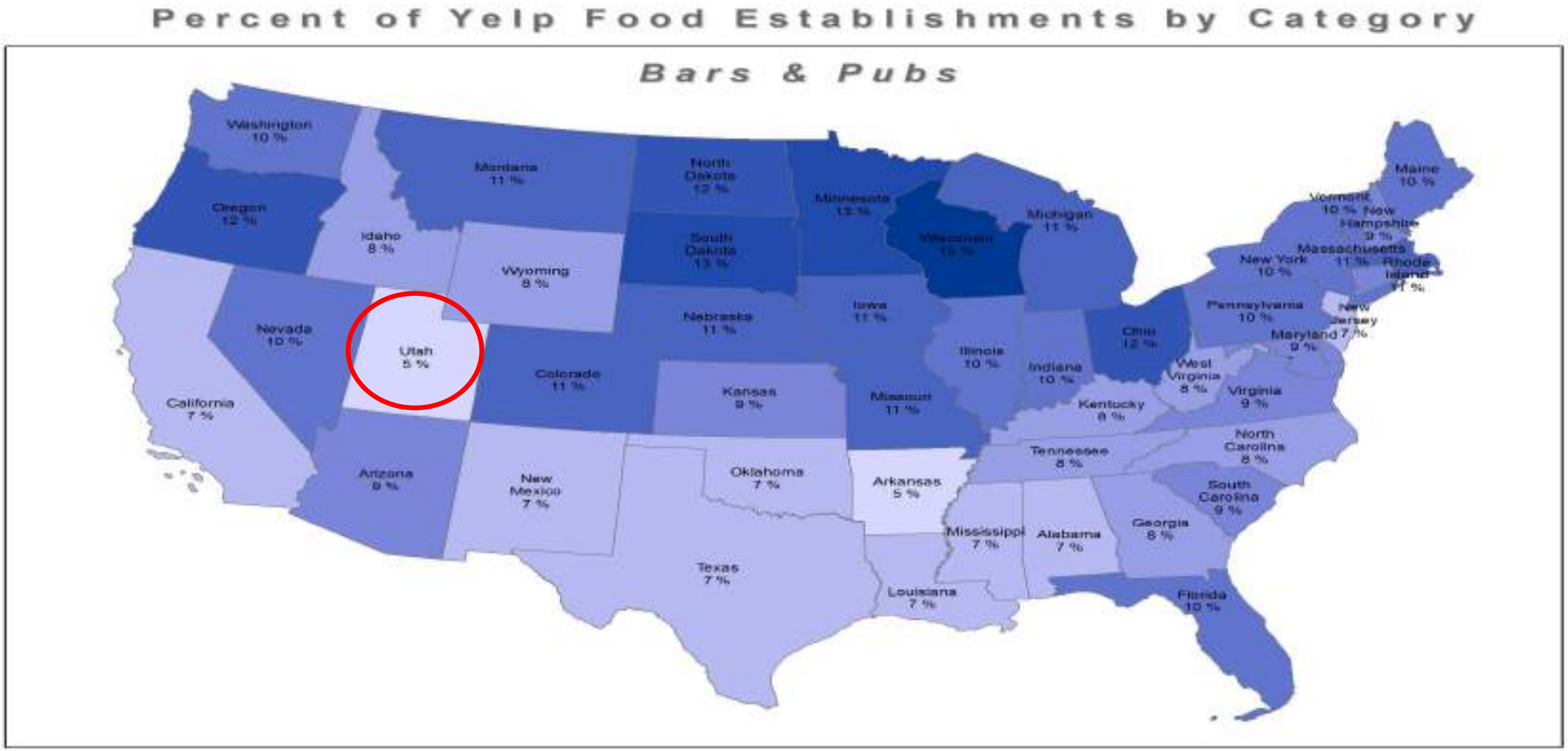


Election Result

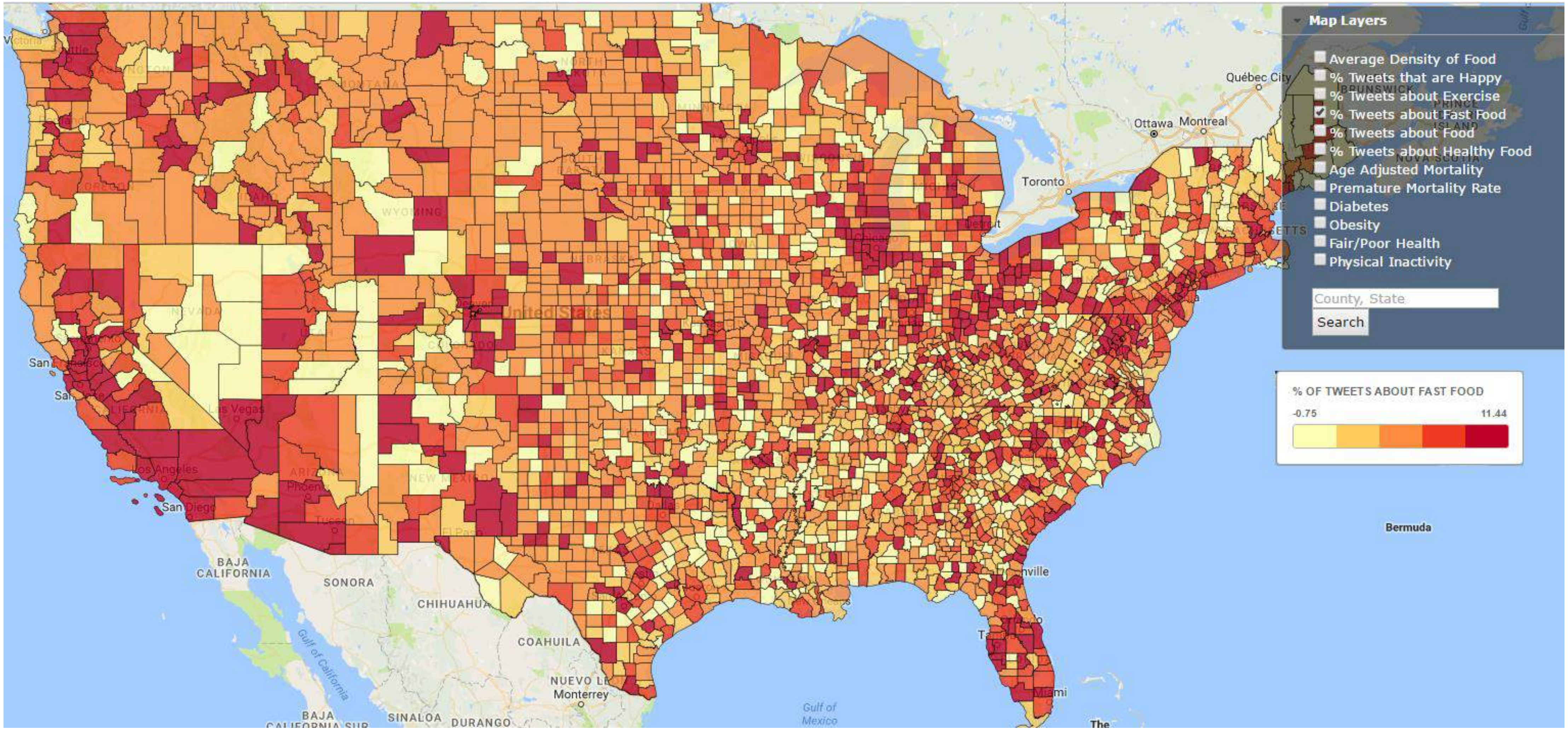
Geotagged based spatio-temporal Sentiment Analysis

California

Application 2: Neighborhood Health Indicator from Social Media Data



Neighborhood Health Indicator from Social Media



April 2015– March 2016. County summaries were derived from 80 million geotagged tweets from the contiguous United States. **https://hashtaghealth.github.io/countymap/map.html**