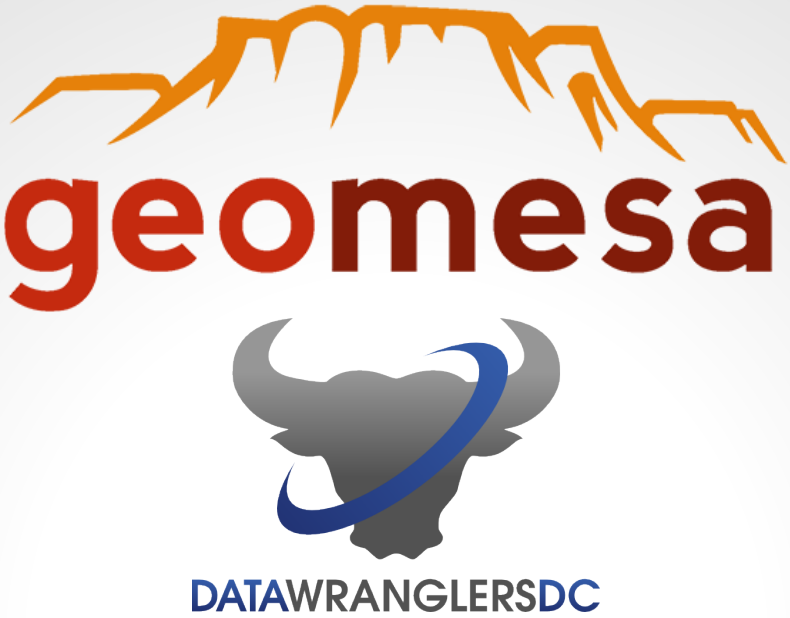


Scaling Spatial Analytics

Andrew Hulbert
ahulbert@ccri.com

Hunter Provyn
hunter@ccri.com

Agenda

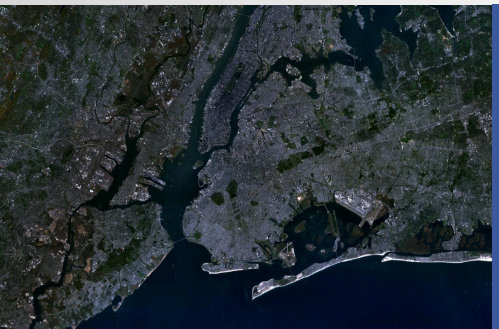
- ➔ Using Spatial Data
- ➔ GeoMesa: Scaling up
- ➔ Spatial Analytics

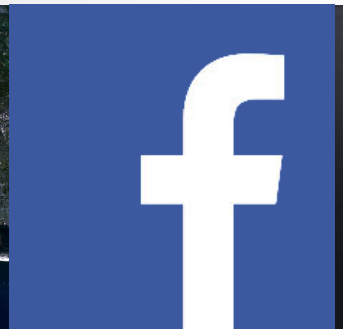
Agenda

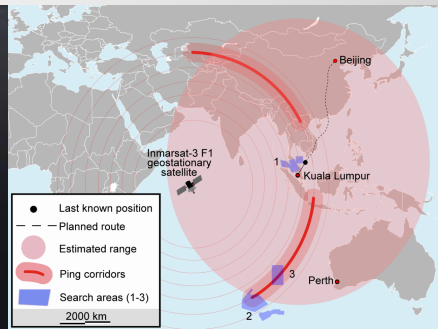
- ➔ **Using Spatial Data**
- ➔ GeoMesa: Scaling up
- ➔ Spatial Analytics

SpatioTemporal Data

- Satellite Imagery
- FAA Flight Information
- Twitter & Social Media
- GPS-Enabled Apps
- Network Traffic & Clickstreams







Questioning Your Data

Who?

What?

When?

Where?

Why?

How?

Questioning Your Data

SpatioTemporal Analytics

Who?

What?

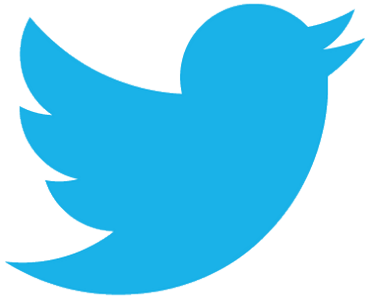
When?

Where?

Why?

How?

Finding Spatial Data



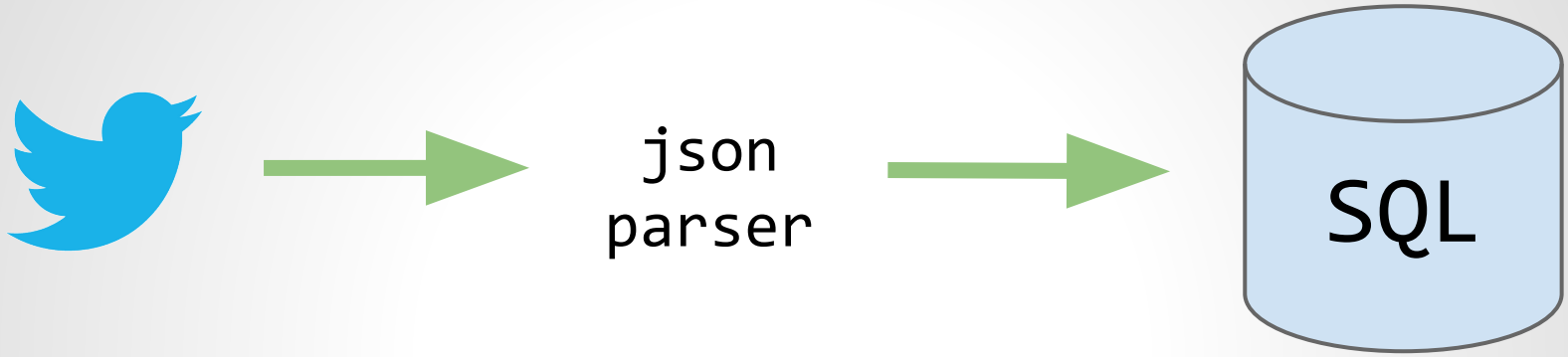
Twitter API

```
{
  "retweeted" : false,
  "id" : 174942523154894848,
  "coordinates" : {
     "coordinates" : [ -75.14310264, 40.05701649 ],
     "type" : "Point"
  },
  "entities" : {
     "hashtags" : [...],
     "user_mentions" : [...],
     "urls" : []
  },
  "created_at" : "Wed Feb 29 19:42:02 +0000 2012",
  "text" : "Man I like me some @twitterapi",
  "user" : {
    "id" : 370773112,
  }
}
```

Found it!

```
{
  "retweeted" : false,
  "id" : 174942523154894848,
  "coordinates" : {
     "coordinates" : [ -75.14310264, 40.05701649 ],
     "type" : "Point"
  },
  "entities" : {
     "hashtags" : [...],
     "user_mentions" : [...],
     "urls" : []
  },
  "created_at" : "Wed Feb 29 19:42:02 +0000 2012",
  "text" : "Man I like me some @twitterapi",
  "user" : {
    "id" : 370773112,
  }
}
```

Wrangling it!



tweet_id	user	lat	lon
0123456	tweeter_man	38.8951	77.0367
9382323	mad_tweeter	39.9500	75.1667
2362363	happy_tweets	40.7127	74.0059
8347347	annoying	41.8369	87.6847

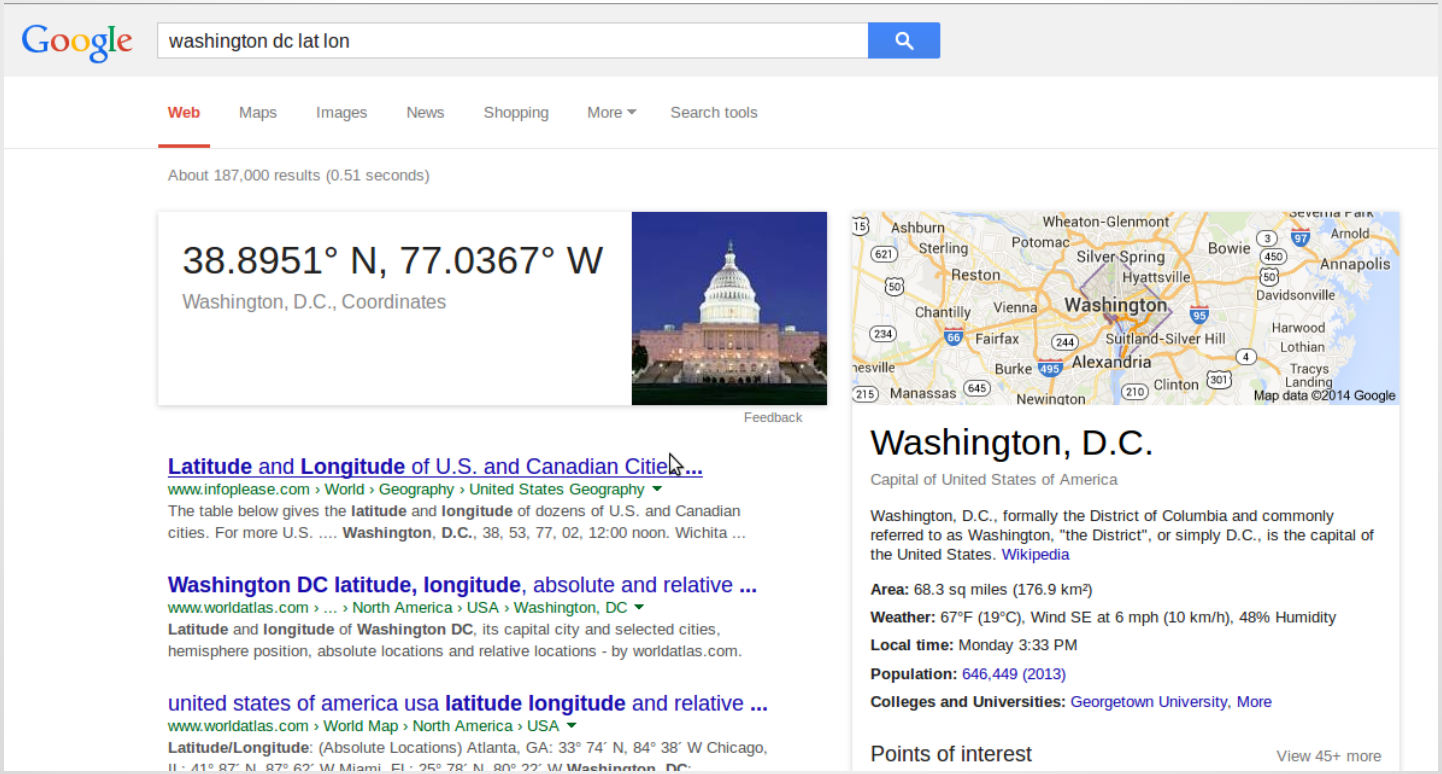
Wrangling it!

tweet_id	user	lat	lon
0123456	tweeter_man	38.8951	77.0367
9382323	mad_tweeter	39.9500	75.1667
2362363	happy_tweets	40.7127	74.0059
8347347	annoying	41.8369	87.6847

Find the tweets in Washington, DC that were re-tweeted at least five times...

Wrangling it!

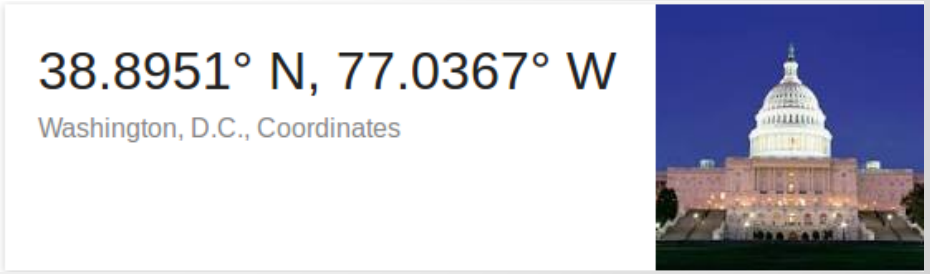
Find the number in Washington, DC that were re-tweeted at least five times...



Source: http://www.google.com

SQL, the sequel to the sequel

```
SELECT * FROM tweets WHERE
   retweet_count > 5
   AND (lat > 38.8 AND lat < 39.0)
   AND (lon > 76.9 AND lon < 77.1)
```



Source: http://www.google.com

Making things harder

Non-point Data

- Roads
- Lakes
- Buildings
- Cities

Source: http://www.openstreetmap.org

Making things harder

Non-point Data

- Lines (road, vehicle track)
- Polygons (lake, building, city)

Hard Queries

- Find all rental properties that border water?
- All cars that traveled within 5 miles of traffic accident hotspots

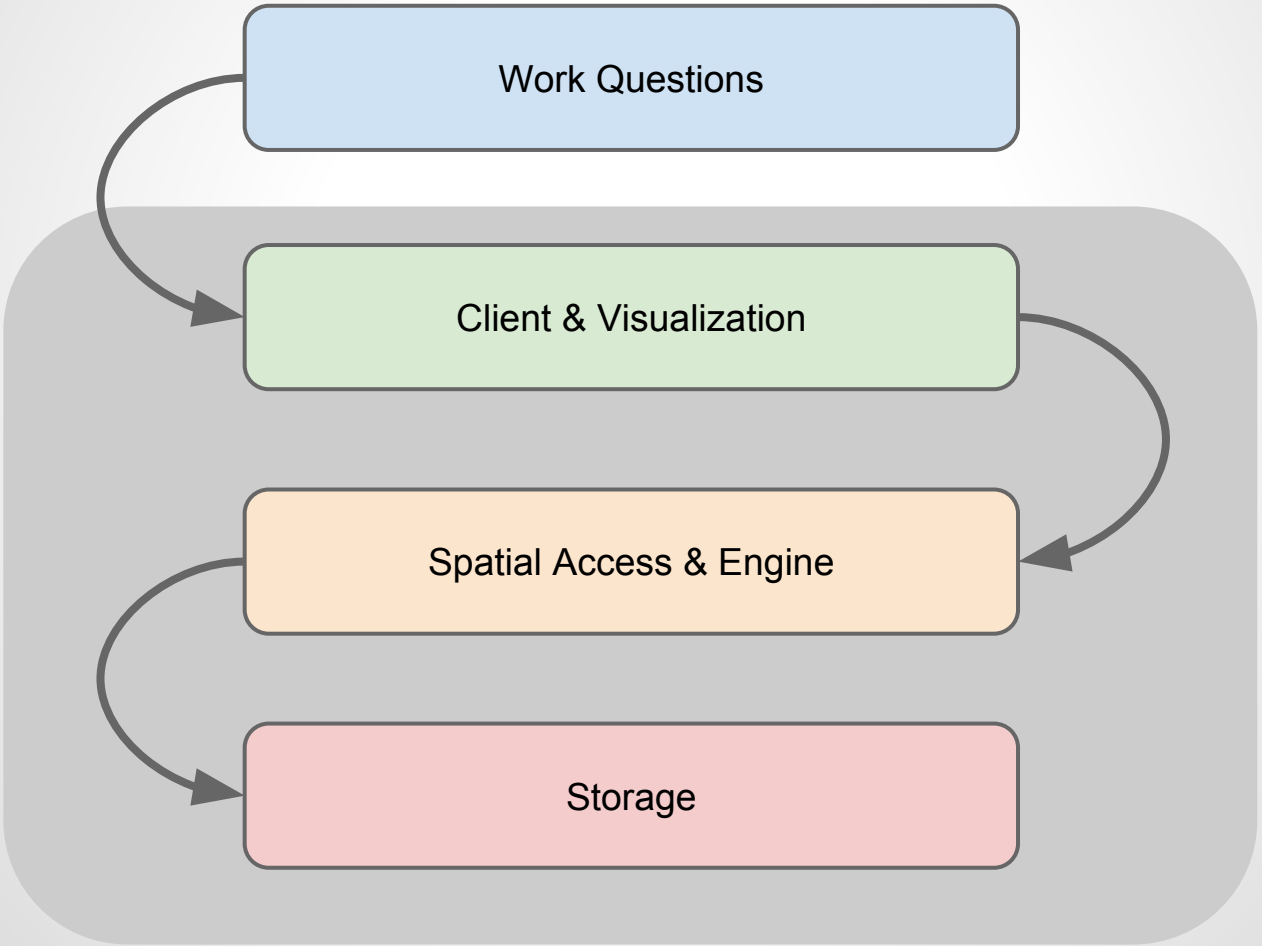
Source: http://www.openstreetmap.org

What do we do?

Source: http://upload.wikimedia.org/wikipedia/commons/3/3b/SteacieLibrary.jpg

Source: http://commons.wikimedia.org/wiki/File:Corinthian_Column_of_the_Temple_of_Zeus_in_Athens.jpg

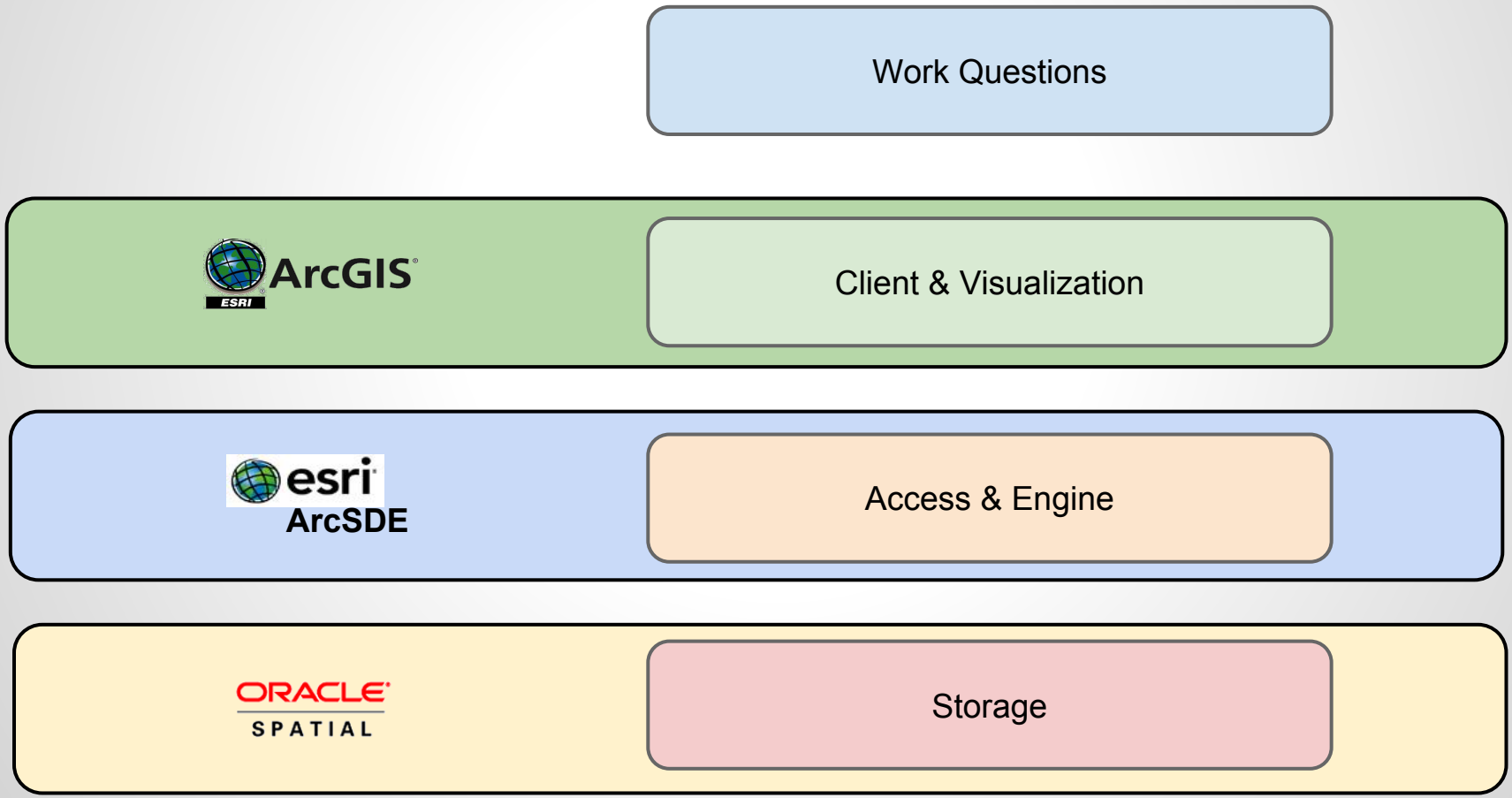
Spatial Analytic Stack



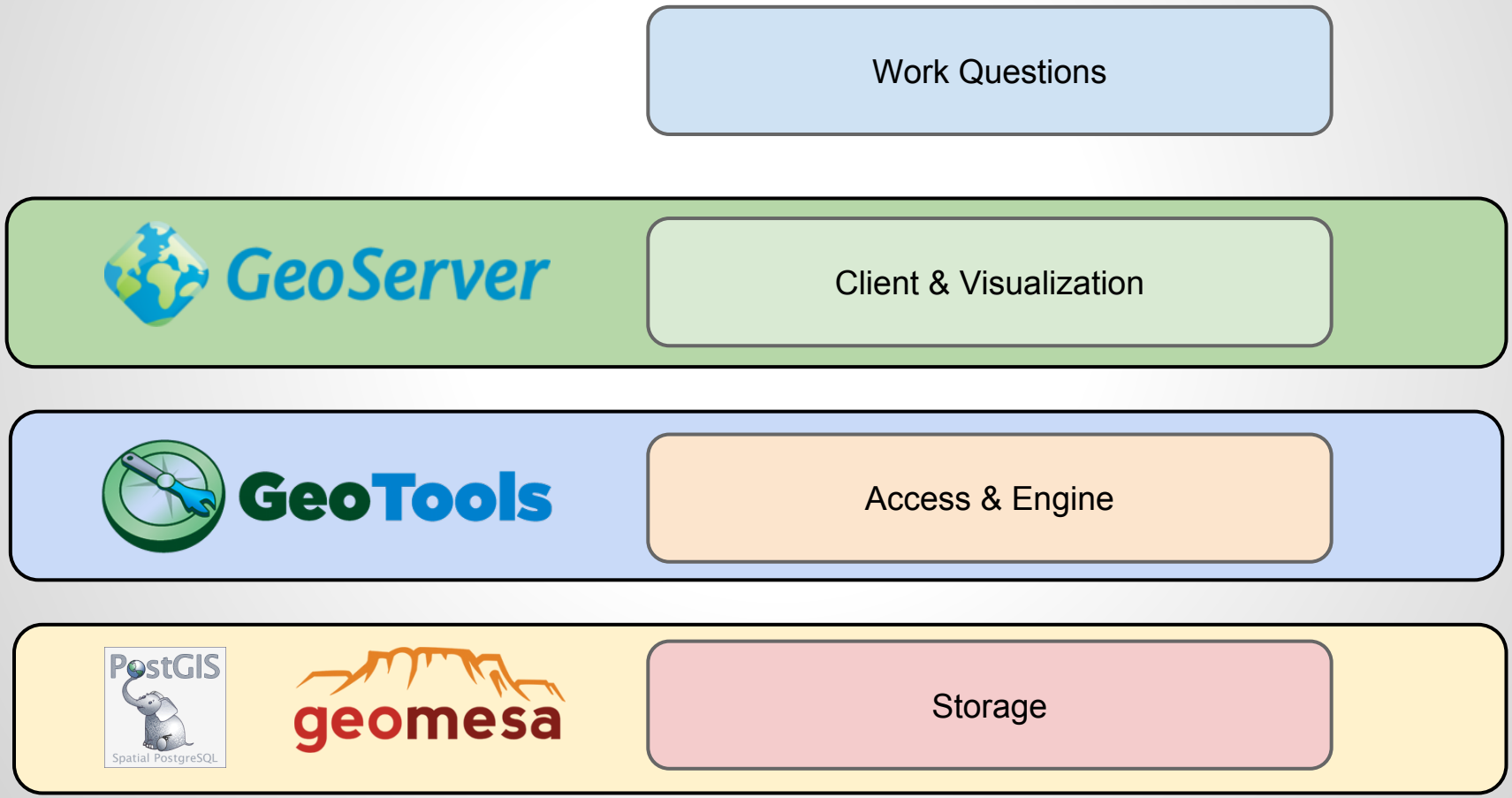
Standards

- Open Geospatial Consortium
  - Geometric Types (Shapes)
  - Web Tier Access & Analytic APIs
- Many Implementations

A Well Known Stack



Open Source Stack



Open Source Libraries

- GeoTools
  - Shapes - Java Topology Suite (JTS)
  - Hard geometric math
- GeoServer
  - J2EE web server
  - OGC API implementations
- OpenLayers
  - JavaScript Implementation
  - Connects to OGC services

Access & Tools

Client & Visualization

Client & Visualization

OGC™ Open Geospatial Consortium, Inc.

GeoTools

GeoServer

Open Source Stack

Work Questions

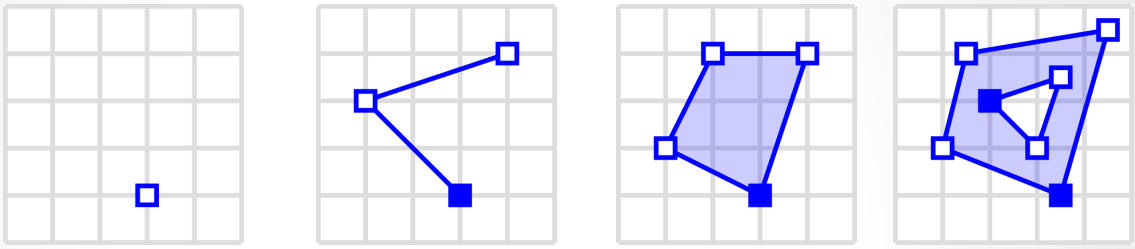
Client & Visualization

Access & Engine

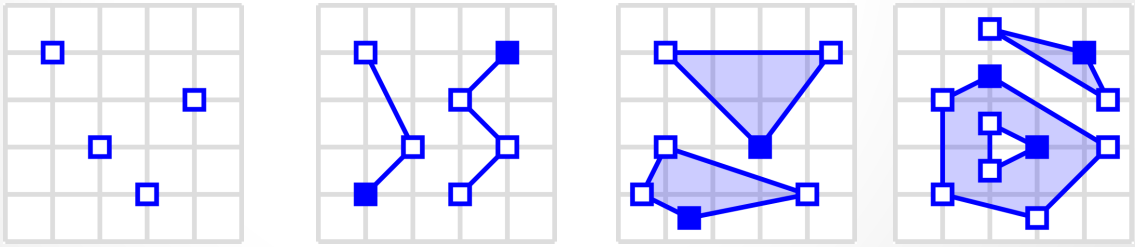
Storage

Spatial Types

Point
LineString
Polygon

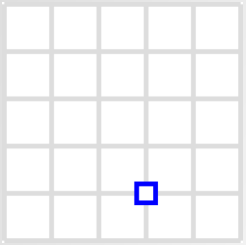


MultiPoint
MultiLine
MultiPolygon

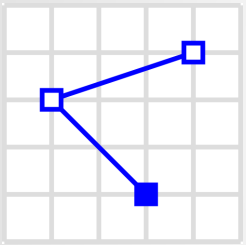


Source: http://en.wikipedia.org/wiki/Well-known_text

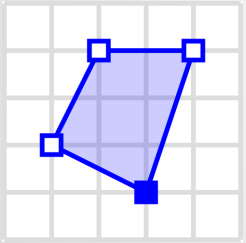
WKT: Well Known Text



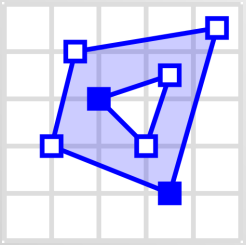
POINT (30 10)



LINESTRING (30 10, 10 30, 40 40)



POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))



POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10),
(20 30, 35 35, 30 20, 20 30))

Source: http://en.wikipedia.org/wiki/Well-known_text

Simple Features: Adding properties


```
{
 "id" : "c11b0bdc-a400-42b3-9177-81f835bff6f5",
 "geometry" : {
   "coordinates" : [-74.189, 40.7728],
   "type" : "Point"
 },
 "type" : "Feature",
 "properties" : {
   "dtg" : "2014-07-08T04:08:43.000+0000",
   "text" : "\"This is our city!\" -David Ortiz",
   "tweet_id" : 01010100101010100101010101001,
   "user_id" : 010101001,
   "user_name" : "Fresh Prince of Belair"
 }
}
```

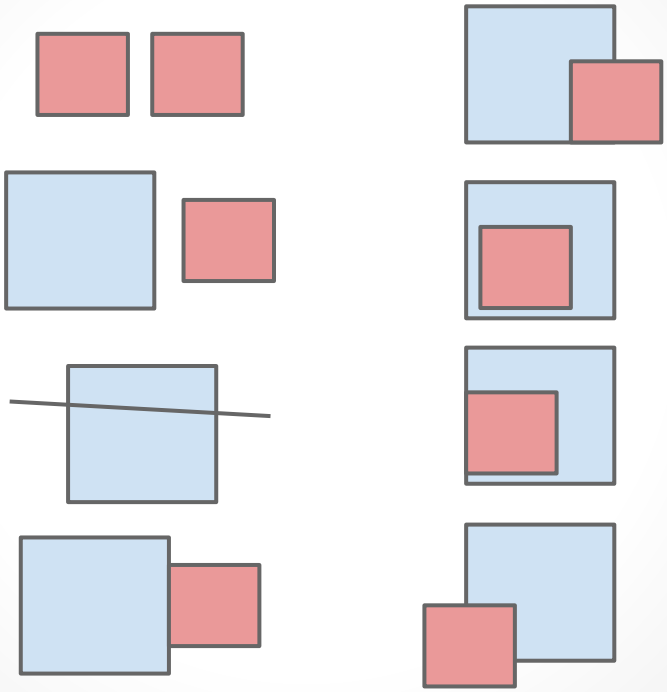
Spatial Relationships

equals

disjoint

intersects

touches



crosses

within

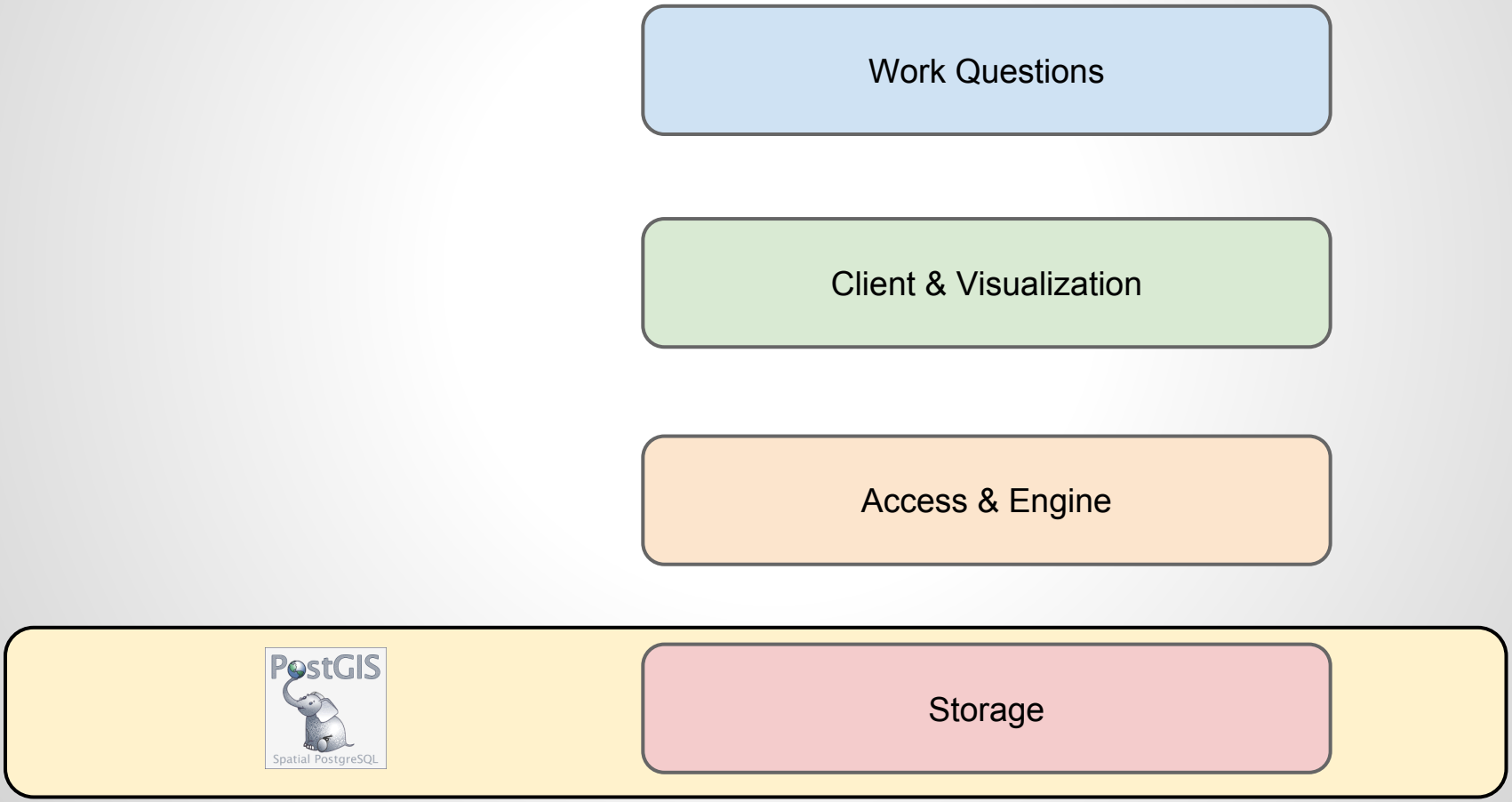
contains

overlaps

GeoTools

VIVID SOLUTIONS

Focusing on Storage



Adapting SQL

```
SELECT * FROM tweets, cities WHERE
   retweet_count > 5
   AND ST_Contains(tweets.geom, city.geom)
   AND cities = “Washington, DC”
```

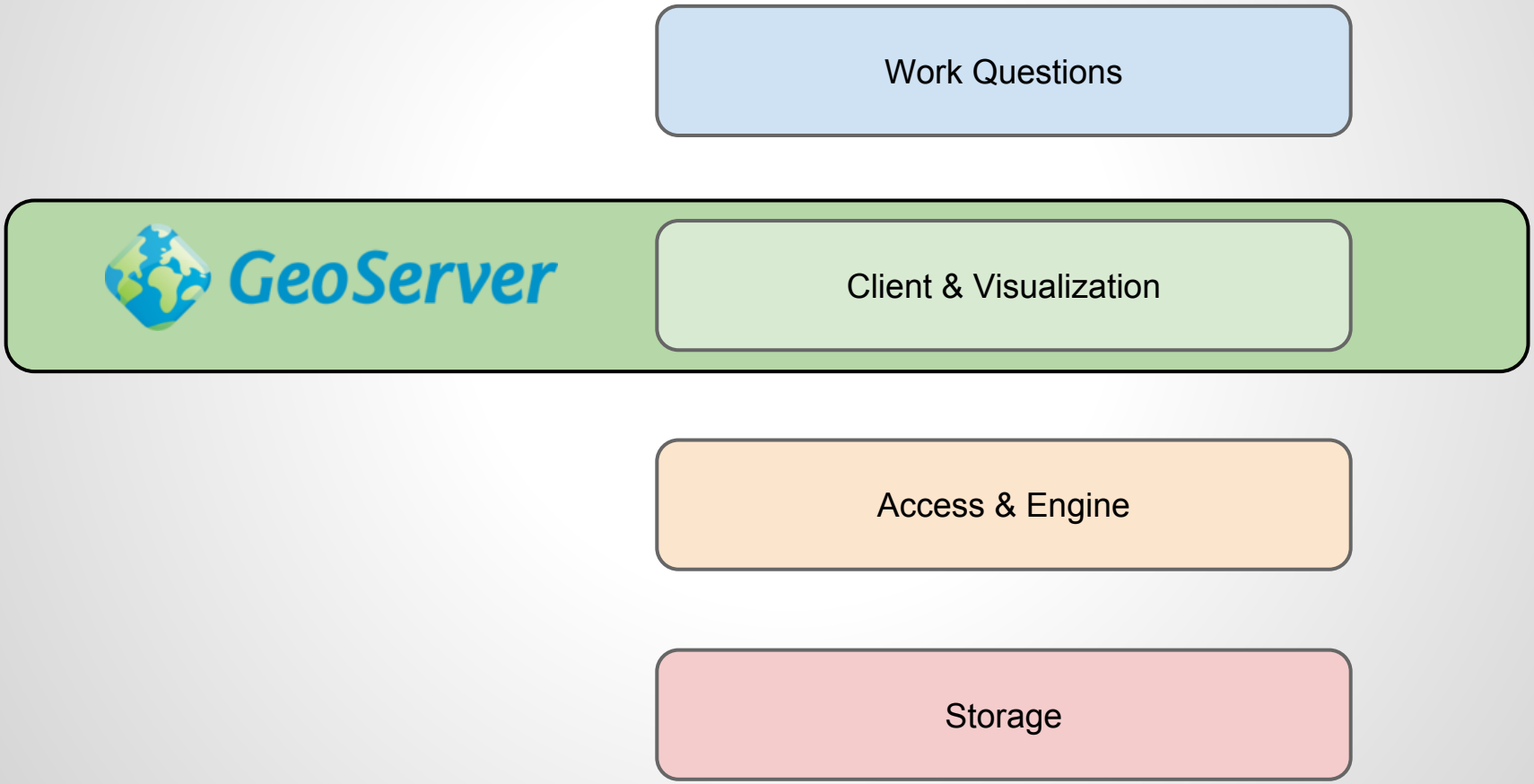
Source: http://postgis.net/

Adapting SQL

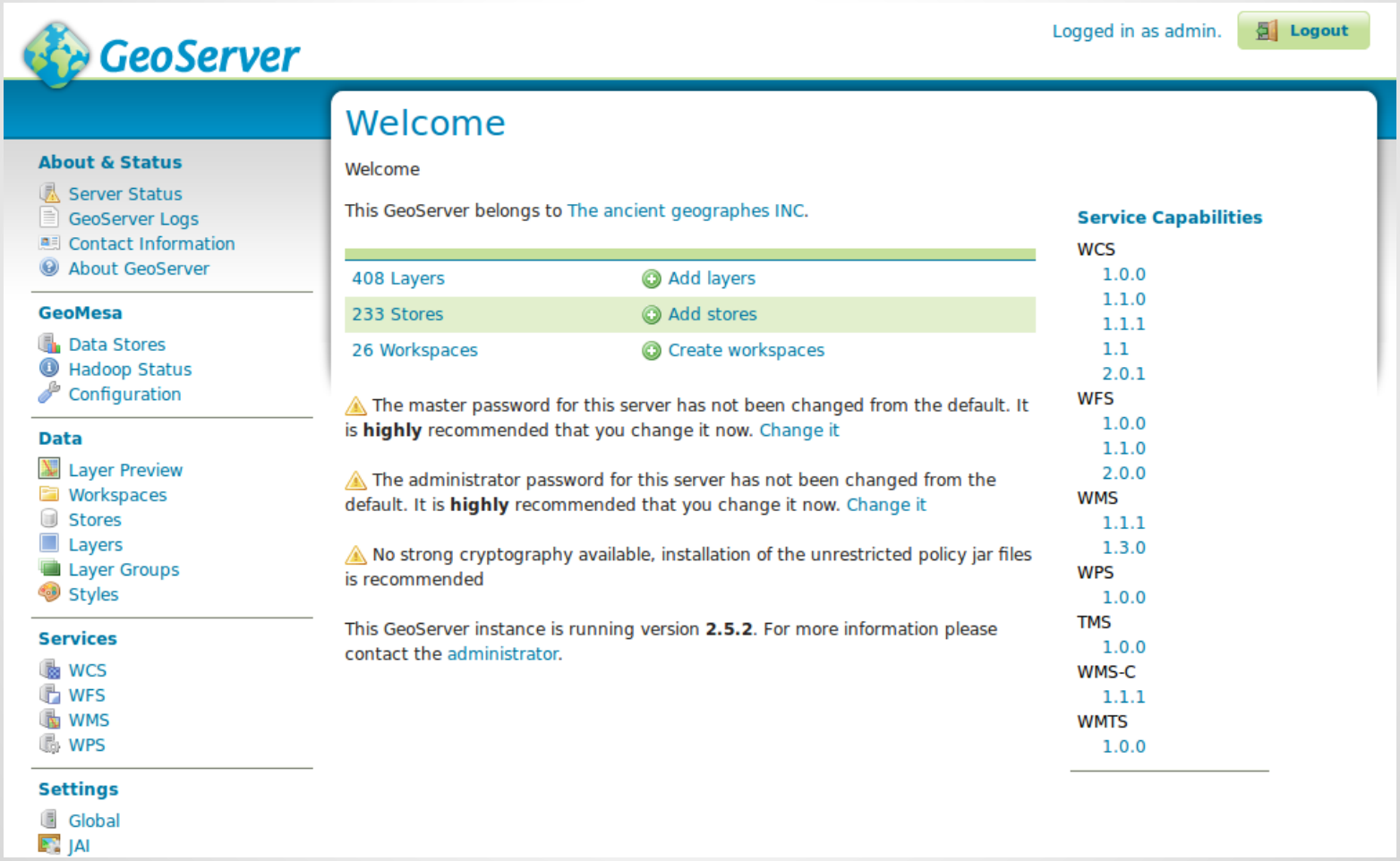
```
SELECT * FROM tweets, cities WHERE
   retweet_count > 5
   AND ST_dwithin(tweets.geom, city.geom, 5000)
   AND cities = “Washington, DC”
```

That’s great...but I like maps

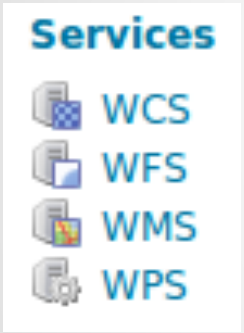
Open Source Stack



GeoServer


GeoServer
Logged in as admin.
Logout
About & Status
Server Status
GeoServer Logs
Contact Information
About GeoServer
GeoMesa
Data Stores
Hadoop Status
Configuration
Data
Layer Preview
Workspaces
Stores
Layers
Layer Groups
Styles
Services
WCS
WFS
WMS
WPS
Settings
Global
JAI
Welcome
Welcome
This GeoServer belongs to The ancient geographes INC.
408 Layers
Add layers
233 Stores
Add stores
26 Workspaces
Create workspaces
The master password for this server has not been changed from the default. It is highly recommended that you change it now. Change it
The administrator password for this server has not been changed from the default. It is highly recommended that you change it now. Change it
No strong cryptography available, installation of the unrestricted policy jar files is recommended
This GeoServer instance is running version 2.5.2. For more information please contact the administrator.
Service Capabilities
WCS
1.0.0
1.1.0
1.1.1
1.1
2.0.1
WFS
1.0.0
1.1.0
2.0.0
WMS
1.1.1
1.3.0
WPS
1.0.0
TMS
1.0.0
WMS-C
1.1.1
WMTS
1.0.0

OGC Services



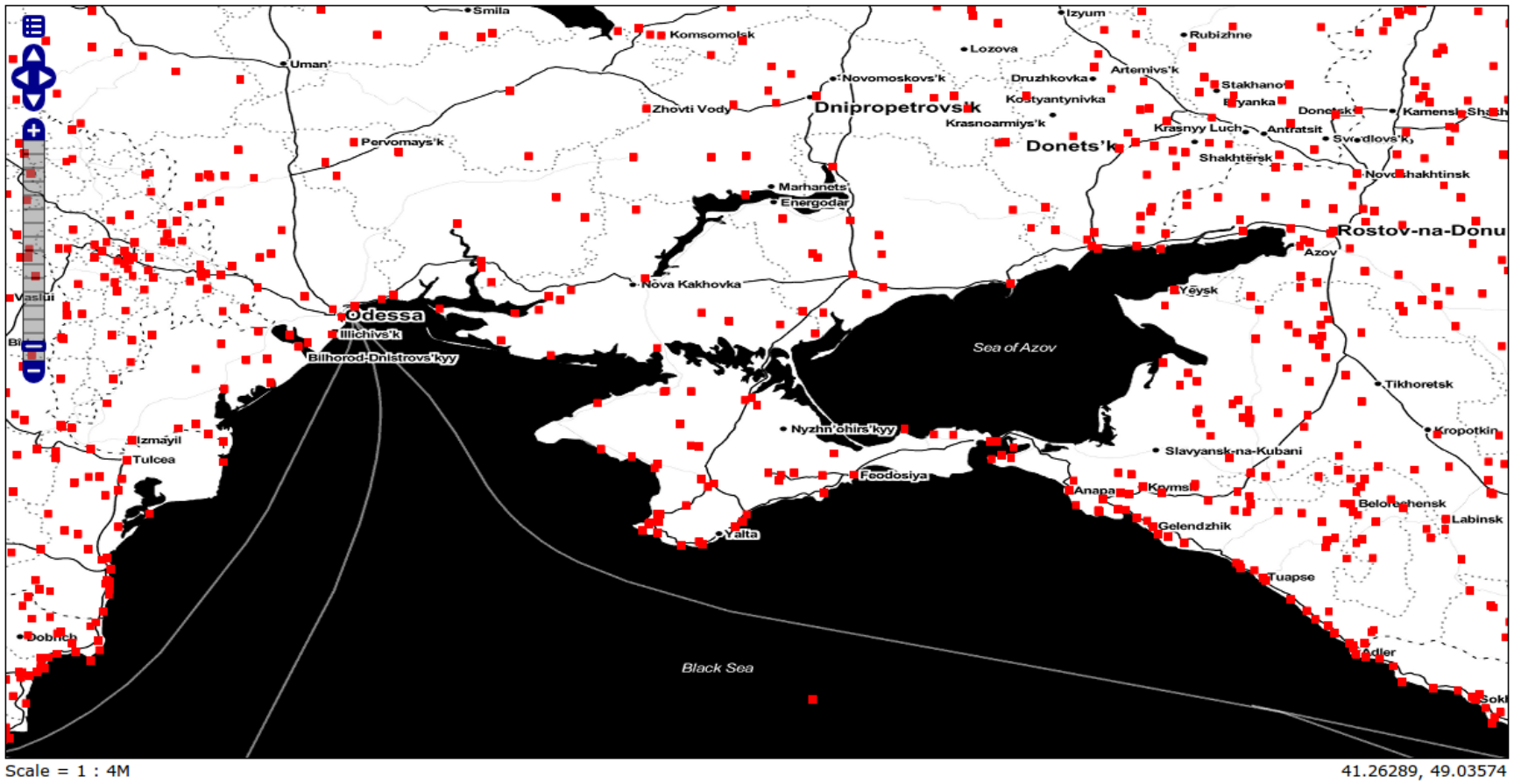
Web Coverage Service

Web Feature Service

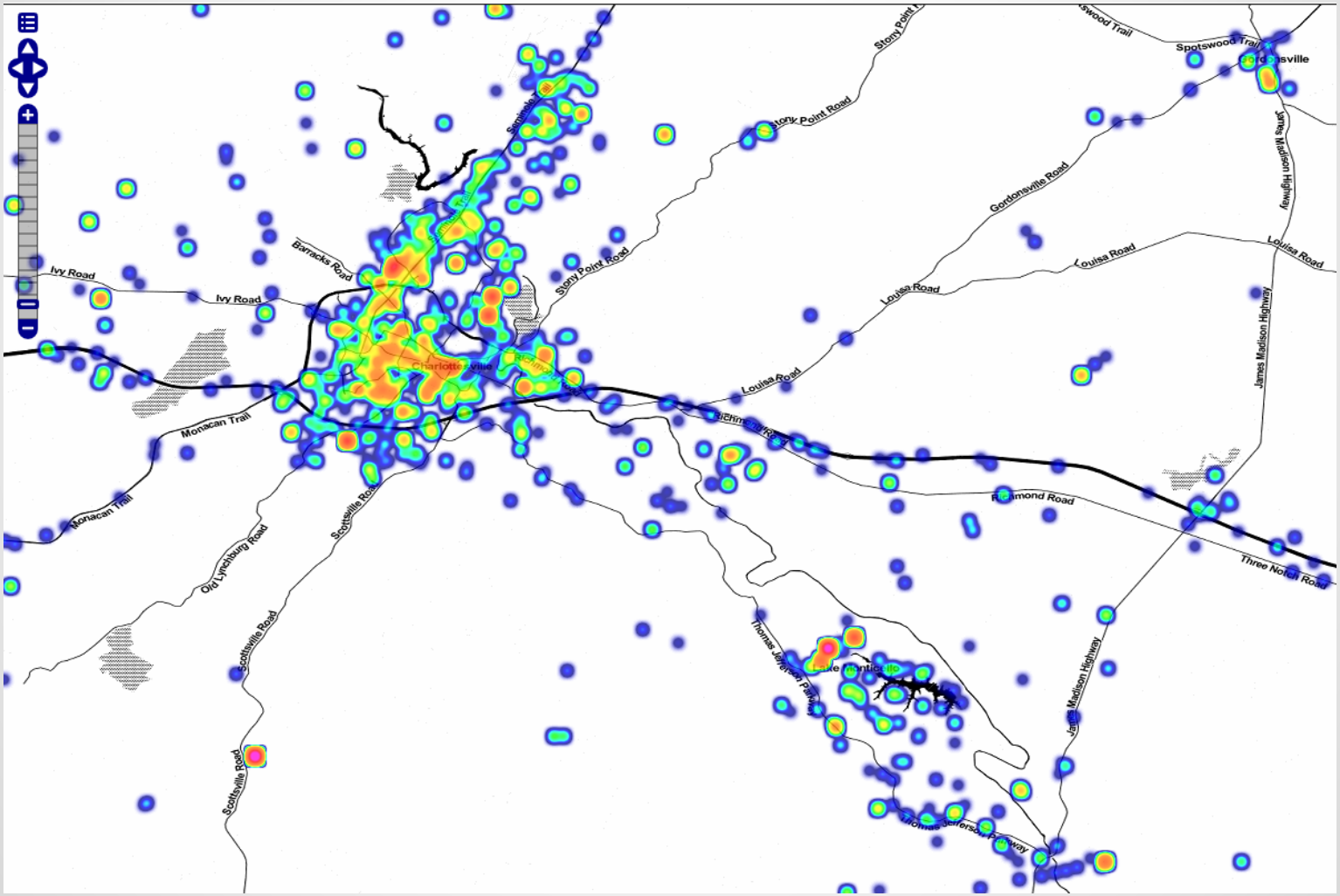
Web Mapping Service

Web Processing Service

Mapping GDELT

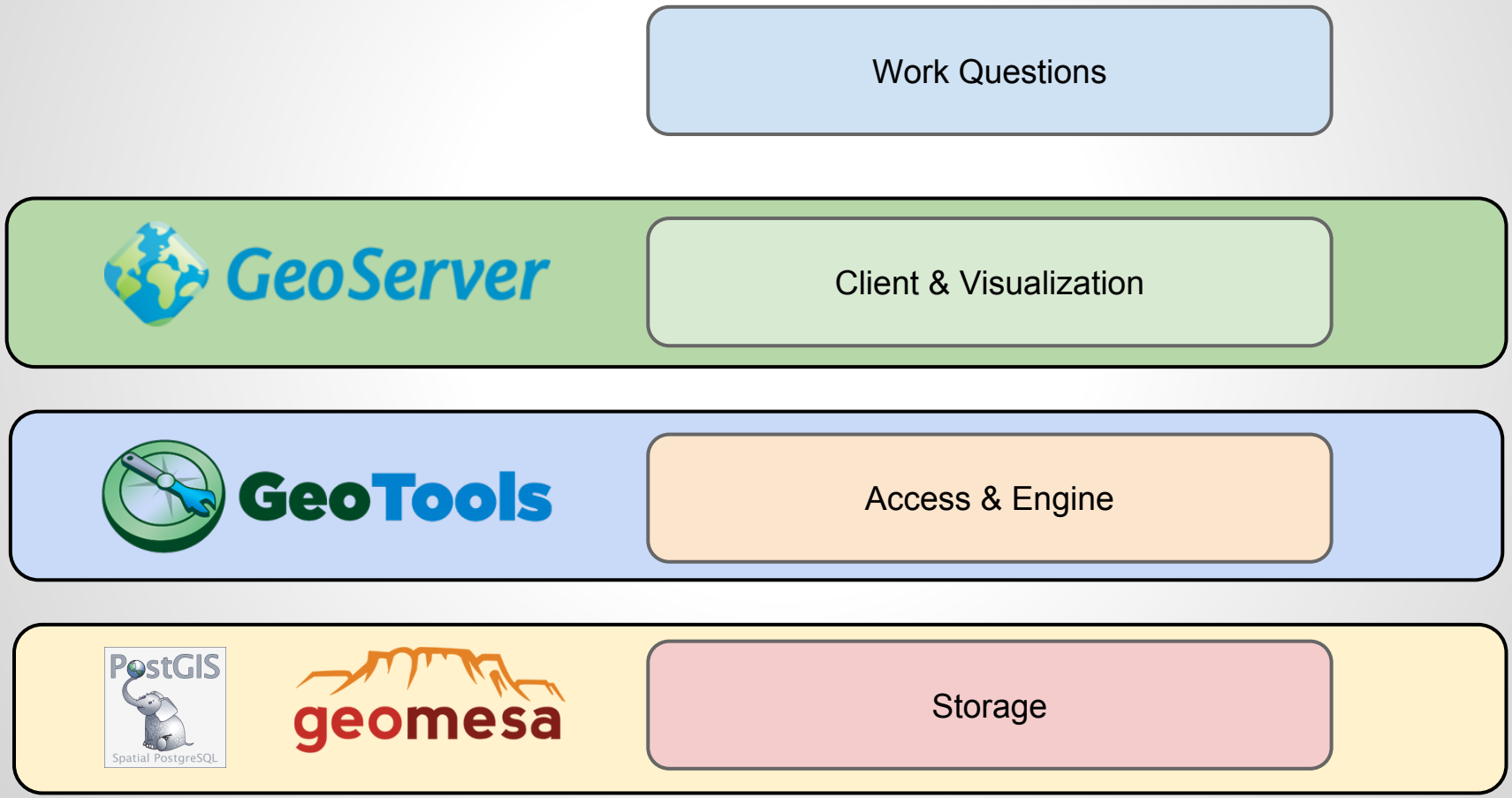


Charlottesville Tweets

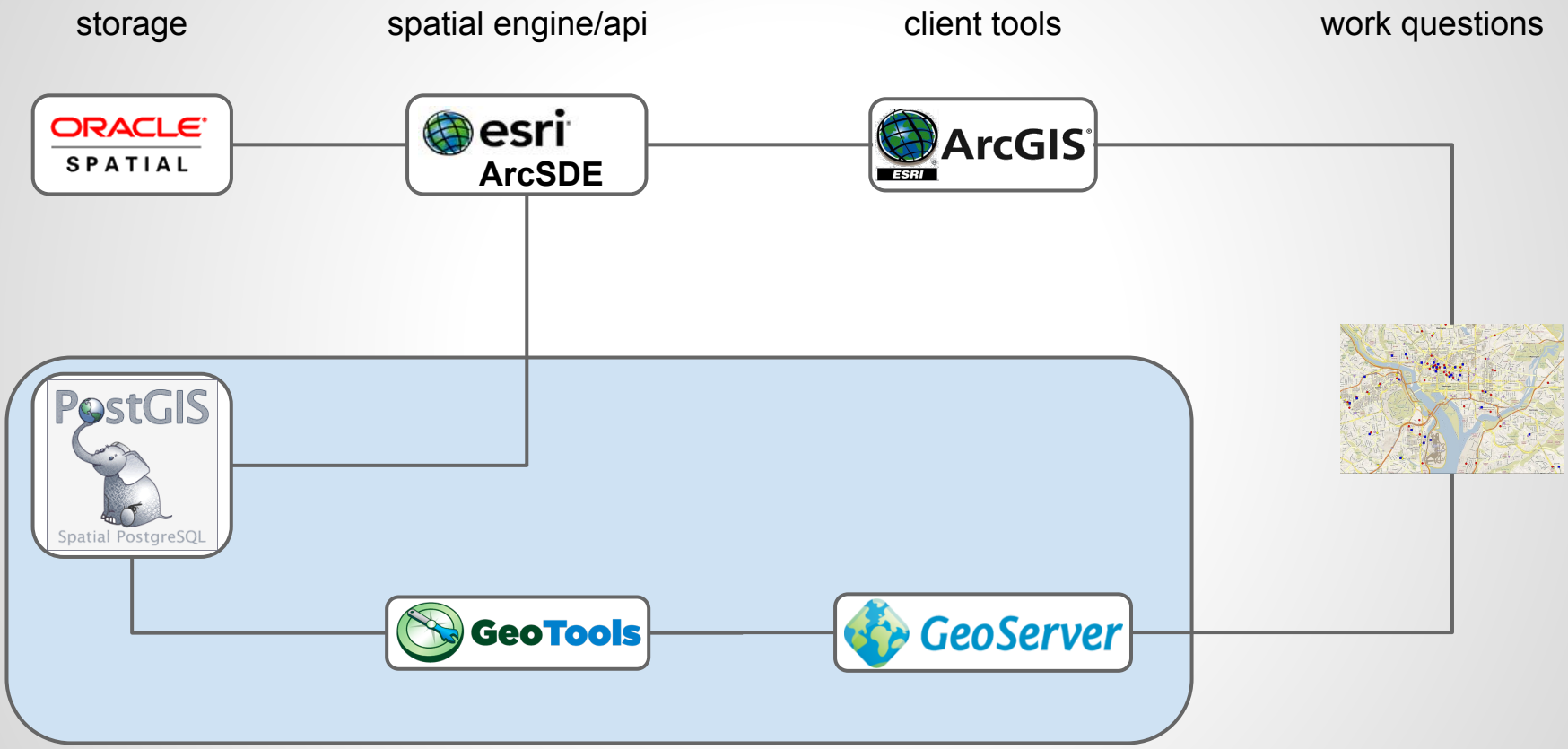


Review: What have we done?

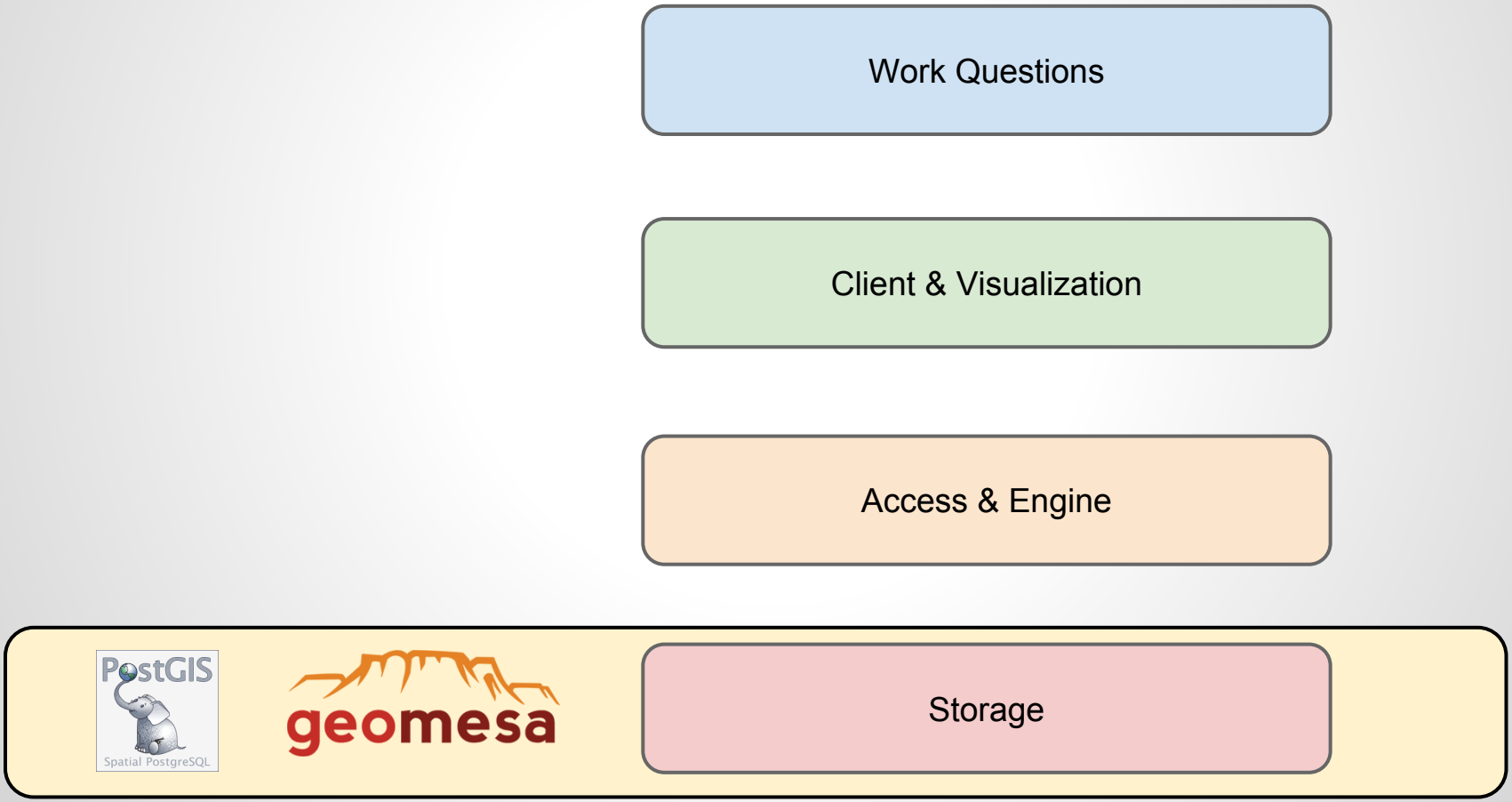
Spatial Analytic Stack



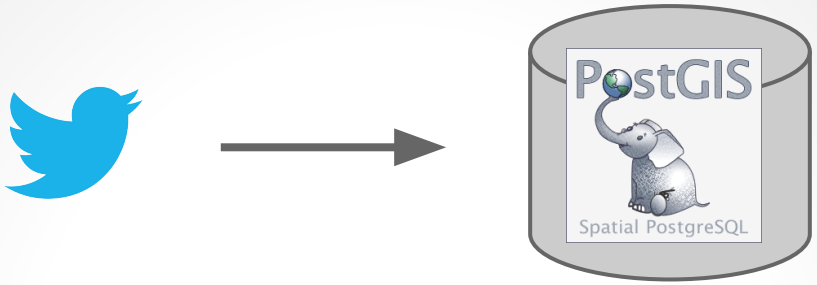
Spatial Analytic Pipelines

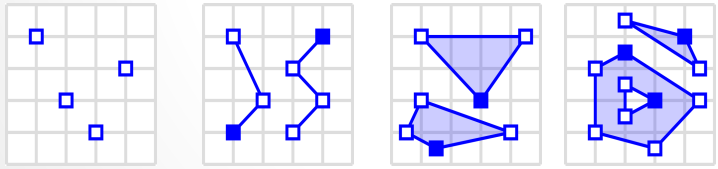


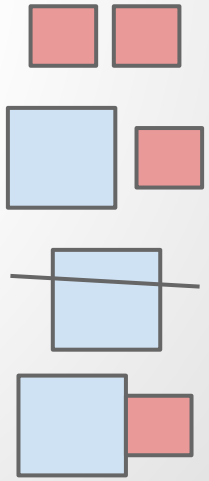
Refocus on Storage



Storing Tweets

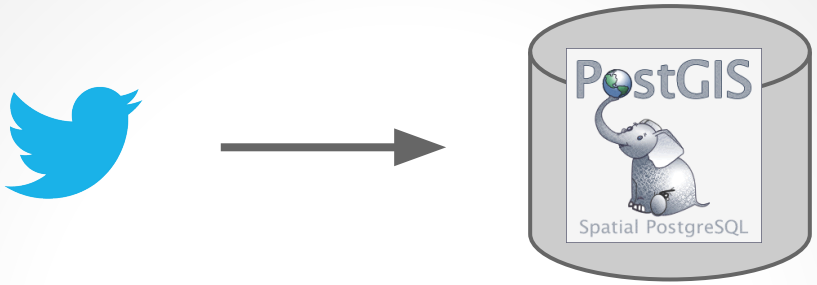


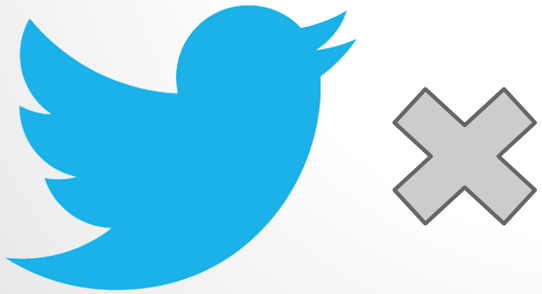




```
ST_dwithin(tweets.geom, city.geom, 5000)
```

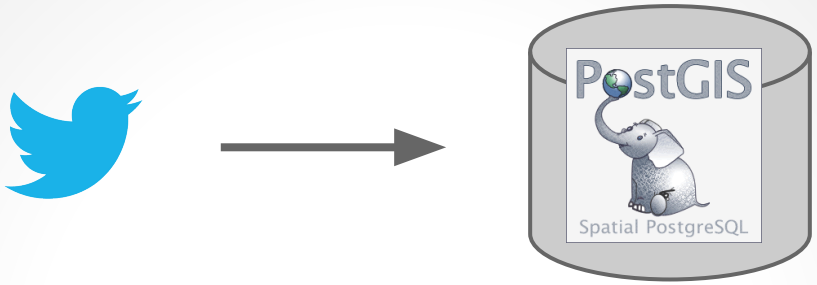
Storing Tweets





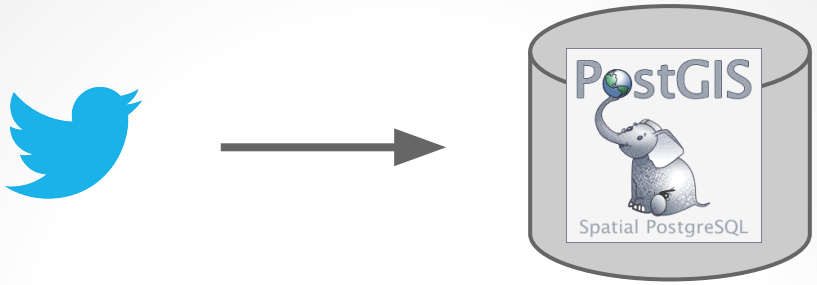
10,000
100,000
1,000,000
100,000,000
1,000,000,000
1,000,000,000,000

Storing Tweets



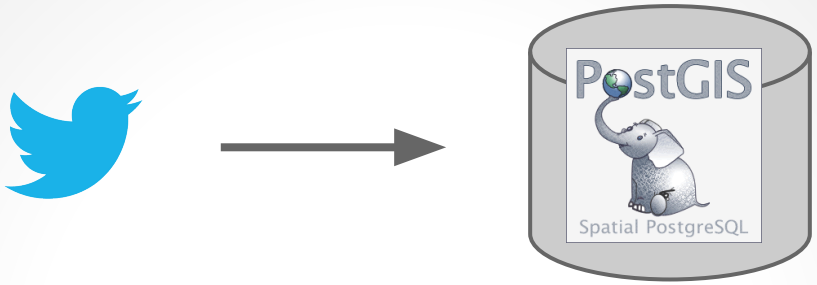
10,000
100,000
1,000,000
one day → **500,000,000**
1,000,000,000
1,000,000,000,000

Storing Tweets



10,000
100,000
1,000,000
one day → **500,000,000** ← 1 PostGIS
1,000,000,000
1,000,000,000,000

Storing Tweets



10,000
100,000
1,000,000
one day → **500,000,000**
1,000,000,000 ← 1 giant SQL DB
1,000,000,000,000

Solutions?

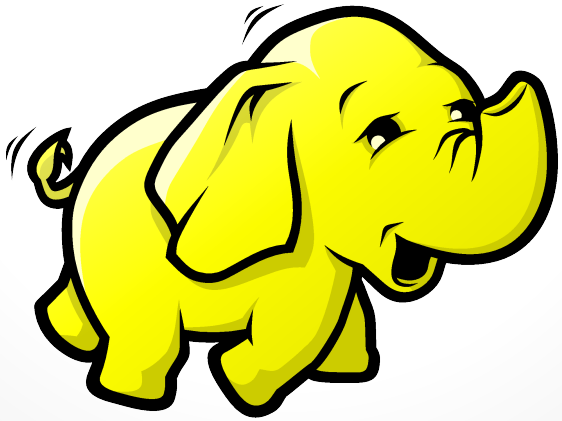
Buy More Databases?

Issues...

- Data Management
- Hardware Cost
- Performance
- Interoperability

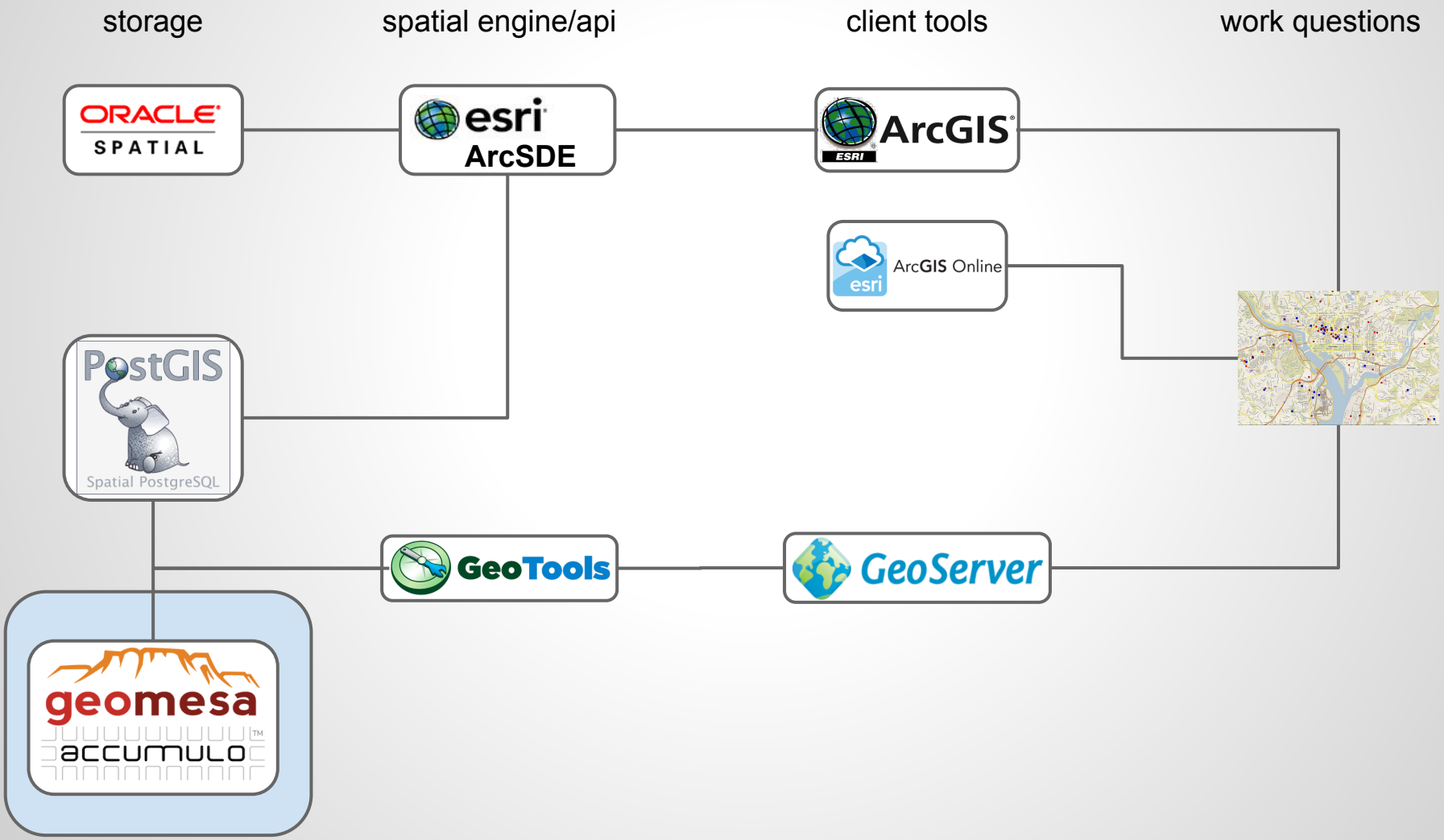
So how do we keep our elephant happy?

Answer: A new Elephant



And his friends...

Spatial Analytic Pipelines



Agenda

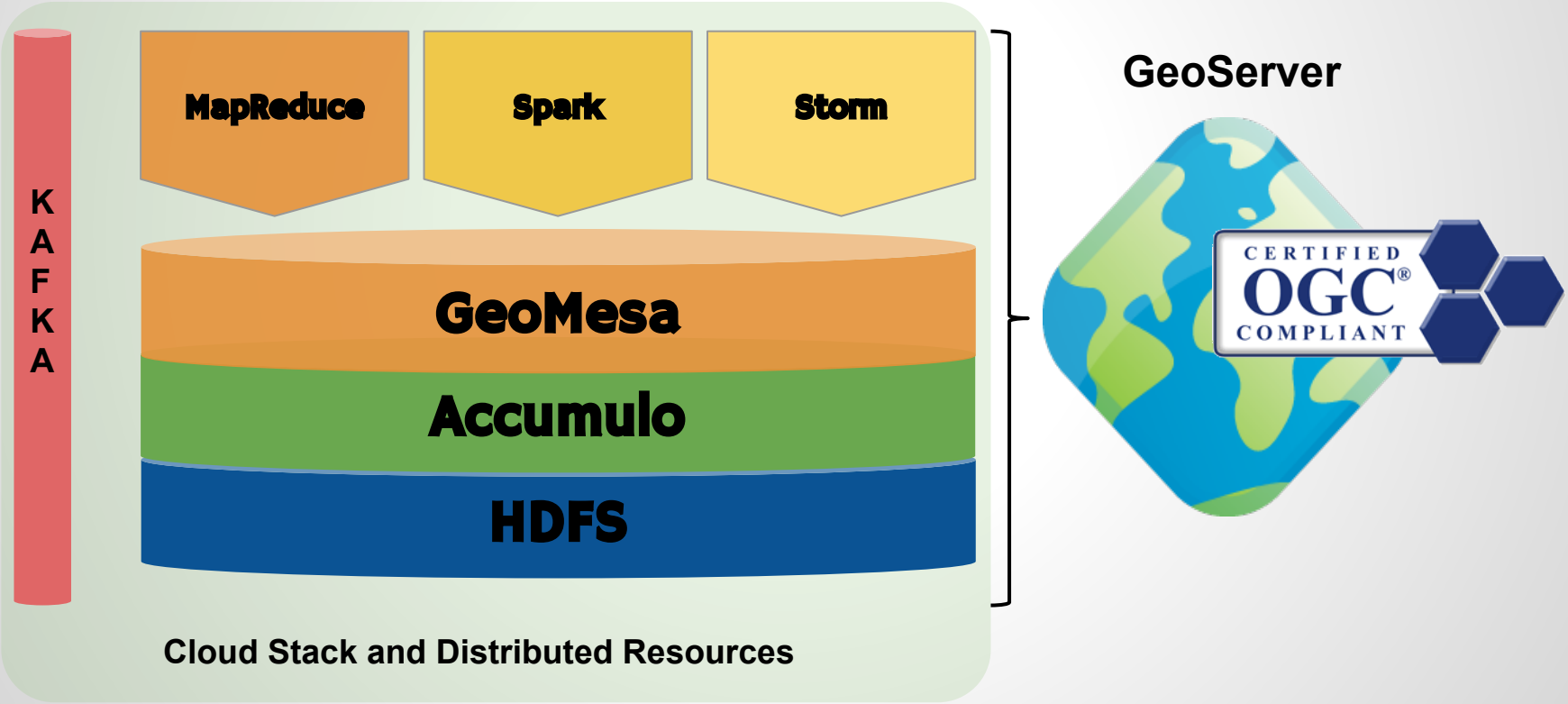
- Using Spatial Data
- **GeoMesa: Scaling up**
- Spatial Analytics

What is GeoMesa?

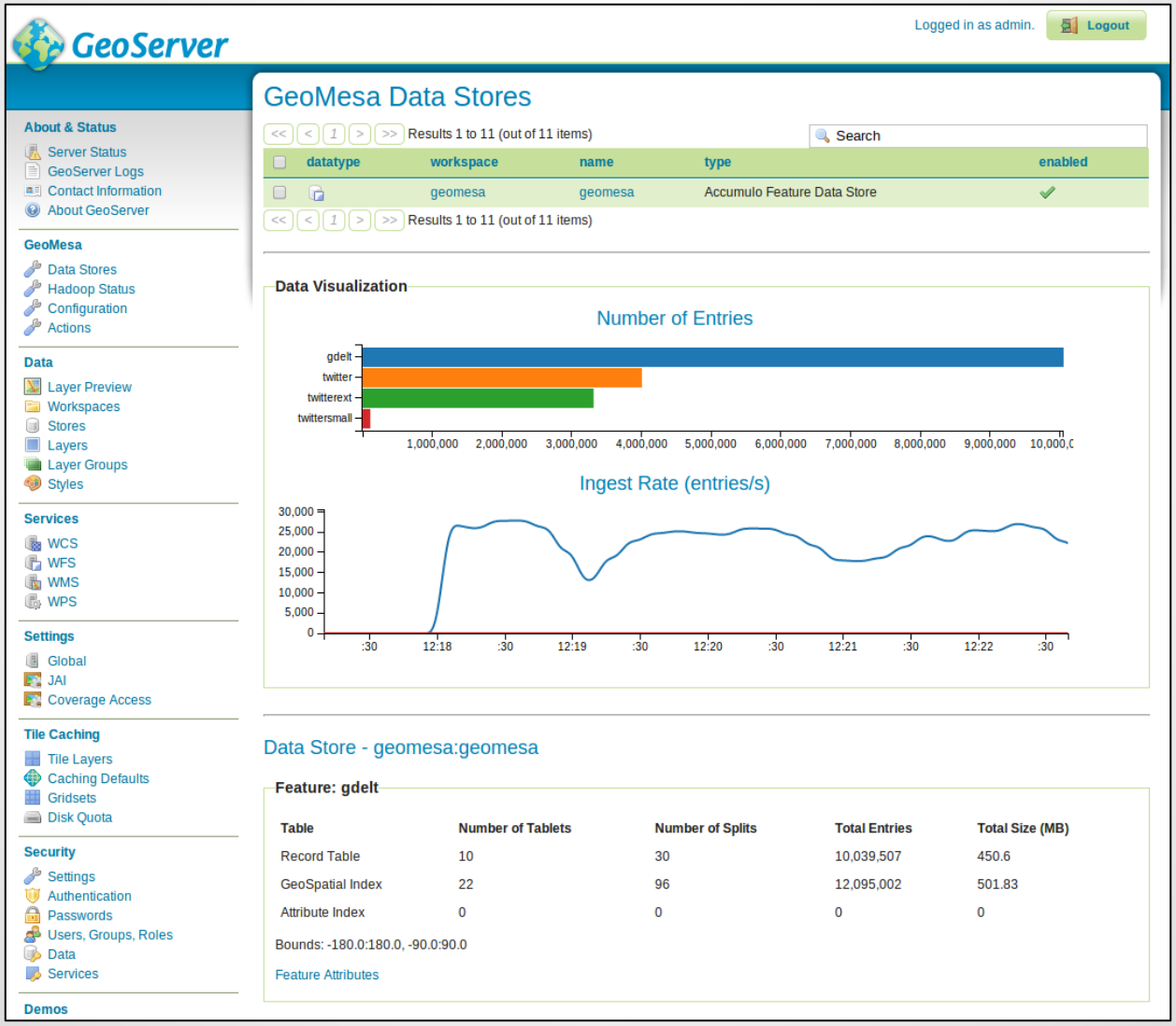
LocationTech

- Advanced, location-aware technologies
- Commercial-friendly open source software
- GeoTrellis, Spatial4j, JTS, UDig and now GeoMesa

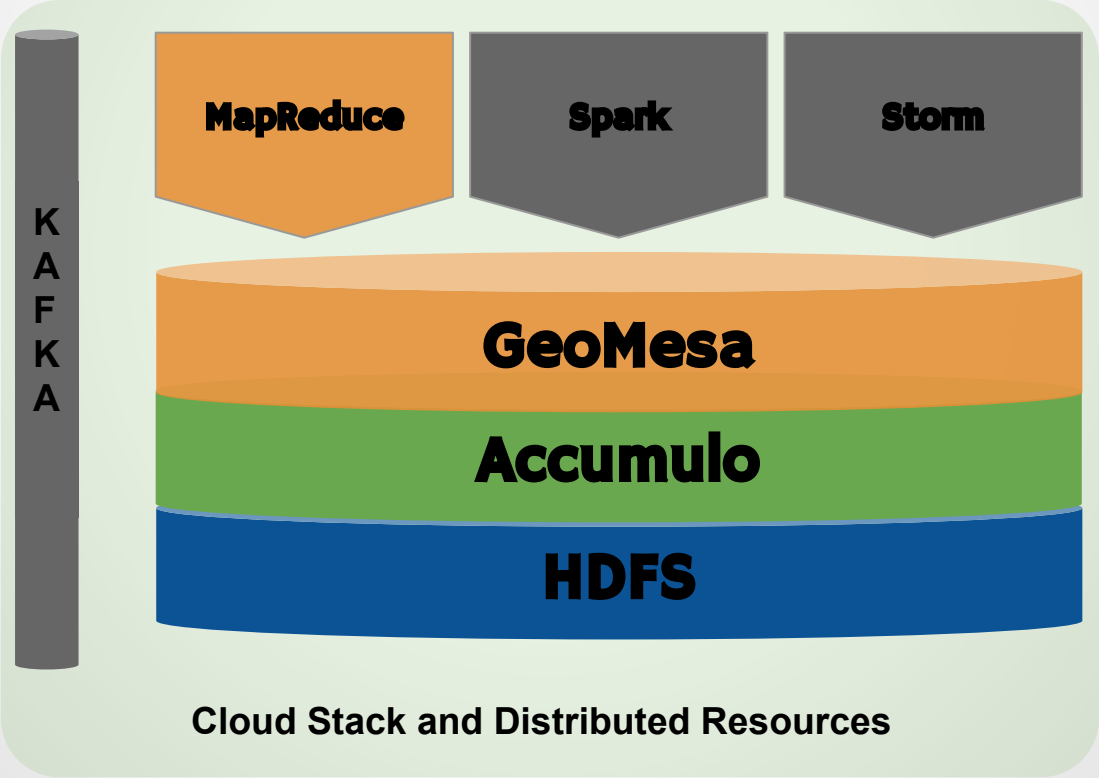
Scalable Open Source Geospatial Stack



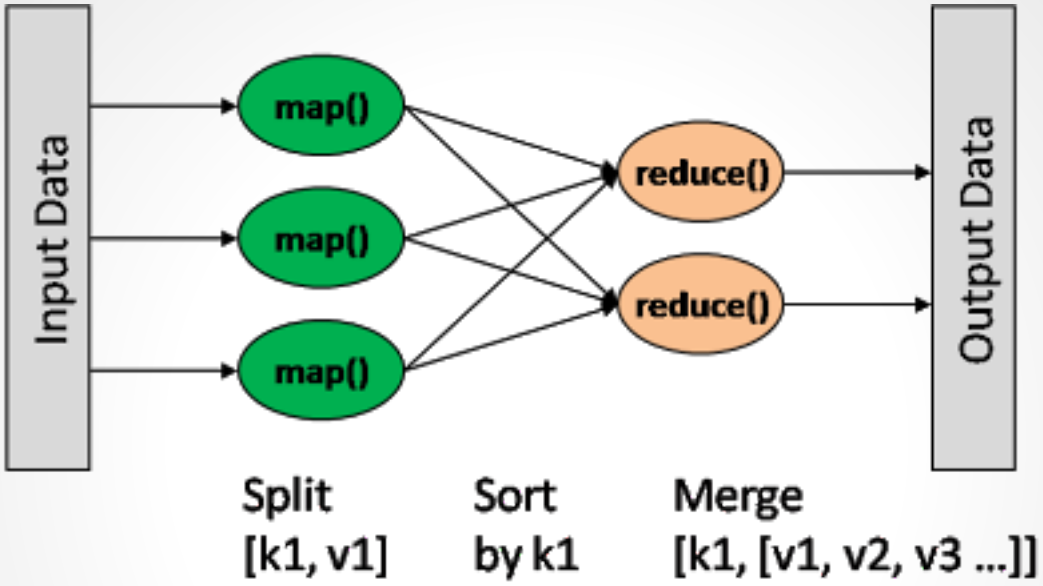
GeoServer Integration



Intro to HDFS + Accumulo Optimized by GeoMesa



HDFS and MapReduce



- Distributed File System
- 128MB blocks
  - establishes data **parallelism**
- One mapper per block

MapReduce Examples

Word count

```
MAP:
foreach line in block:
  words = line.split(SPACE)
  foreach word in words:
    emit(word, 1)

REDUCE:
foreach (word, list[Int]):
  emit(word, list.sum)
```

HeatMap

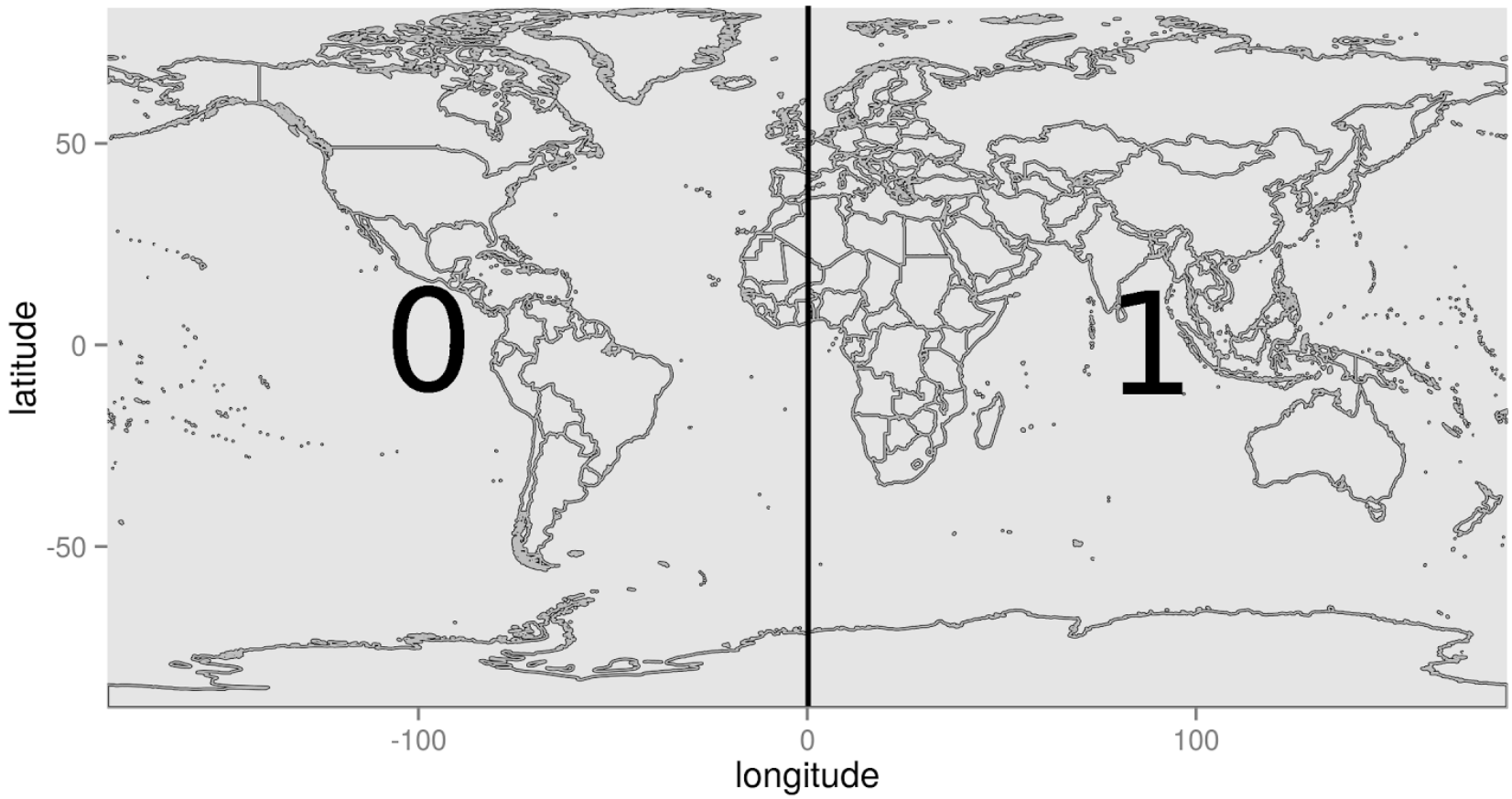
```
MAP:
foreach feature in features:
 pixels = world2screen(feature)
 foreach pixel in pixels:
   emit(pixel, 1)

REDUCE:
foreach (pixel, list[Int]):
  emit(pixel, list.sum)
```

Accumulo

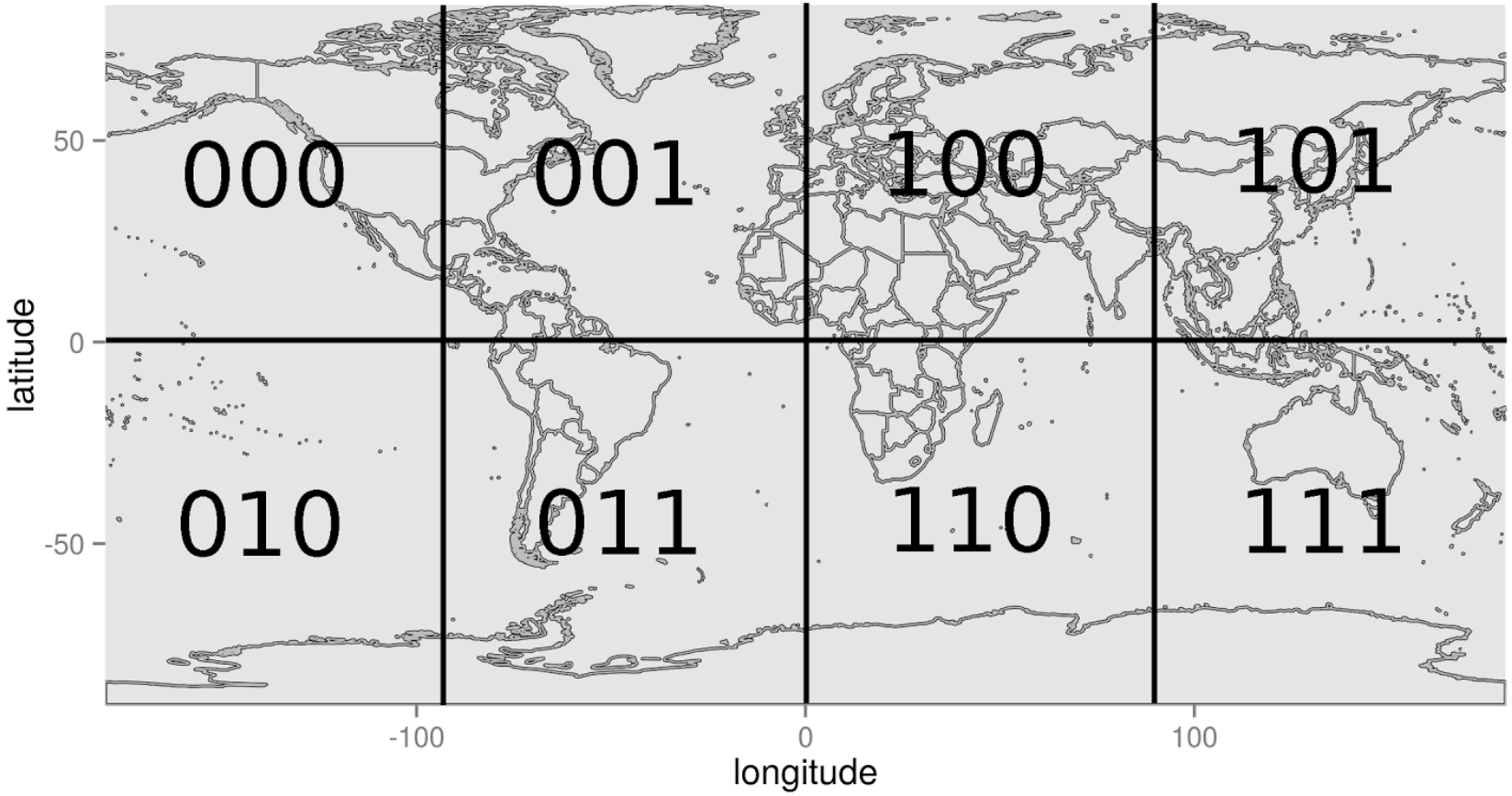
- Distributed Database on HDFS
- Column-oriented Storage / Key Value Store
- Cell-Level Security
- Lexicographic Indexes
  - GeoMesa optimizes it for geospatial
  - GeoMesa(Space and Time) -> single dimension

Space-filling Curve



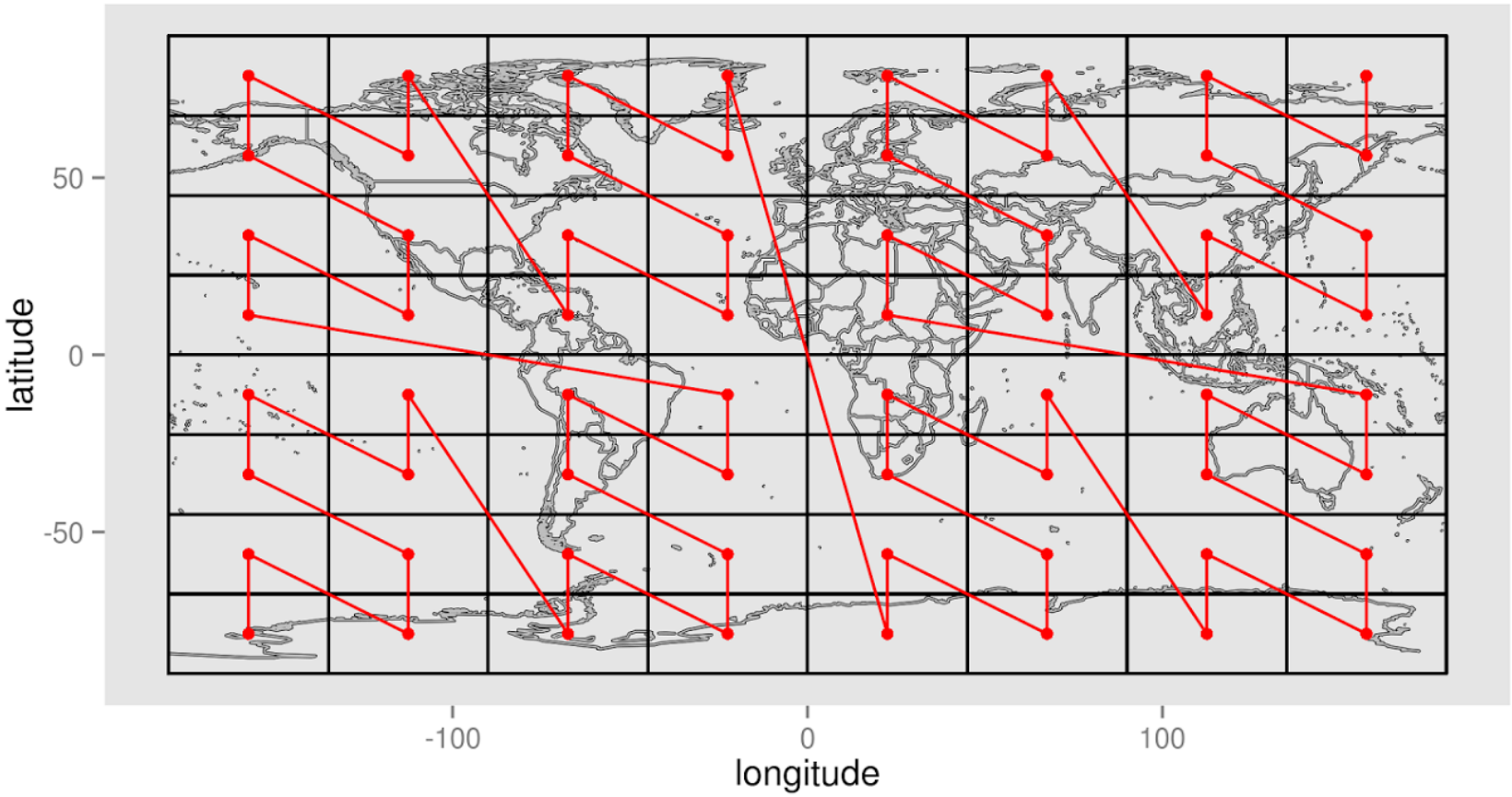
Geohash = interleaved lat long bits

Space-filling Curve



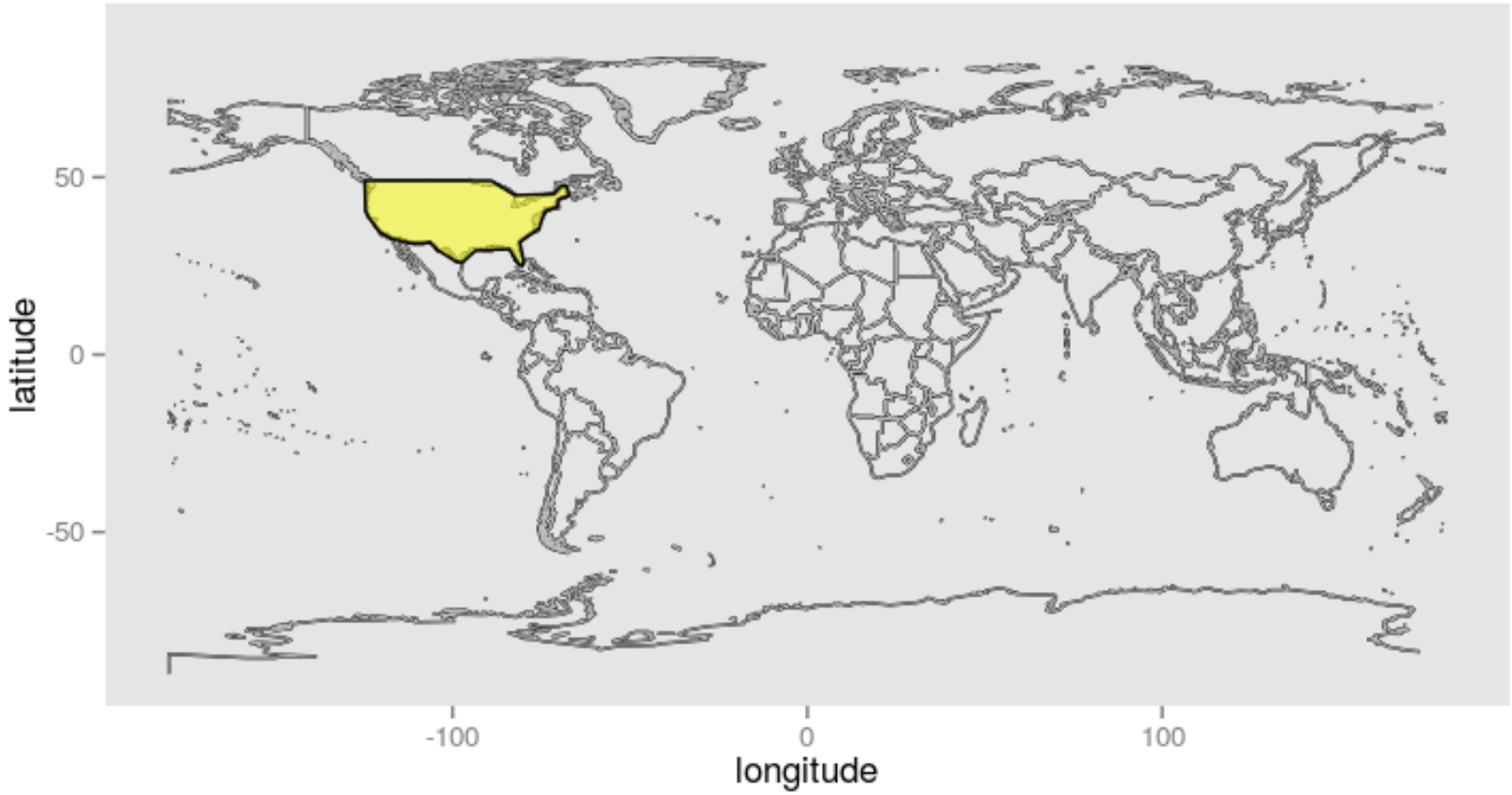
Geohash = interleaved lat long bits

Space-filling Curve



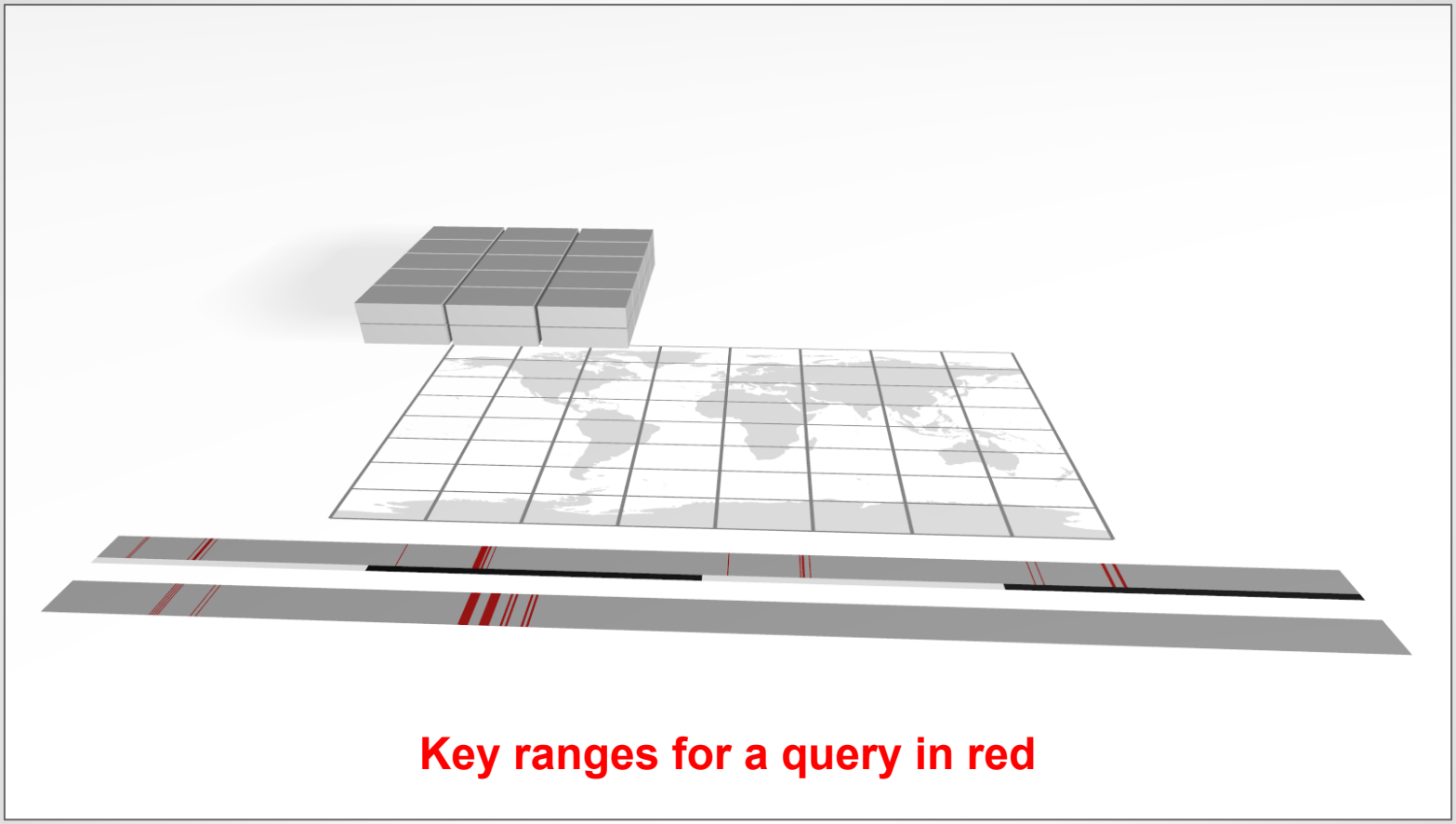
Locality through Geohashes

Spatial Query Planning



Decompose query polygon into GeoHash ranges to scan

Partitioning the Key Space



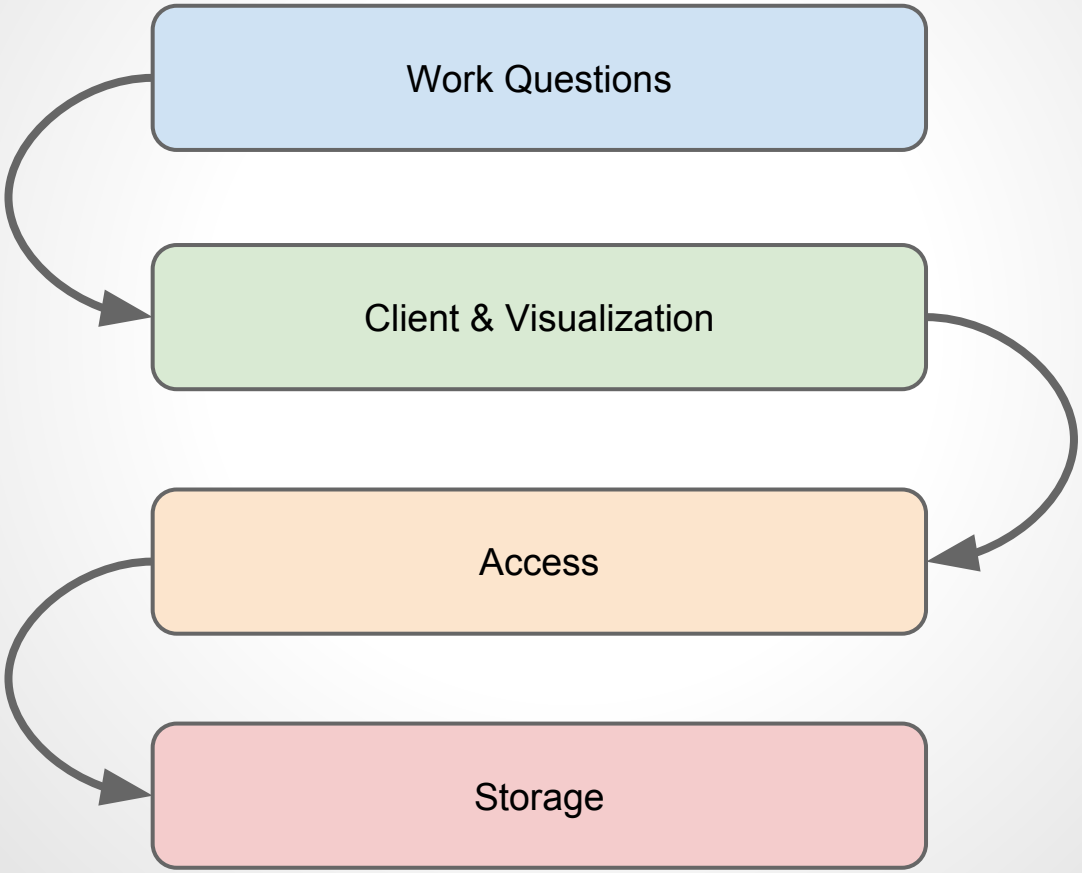
Top row: 4 partitions
Bottom row: No partitions

"gonna make every tablet server sweat"

Agenda

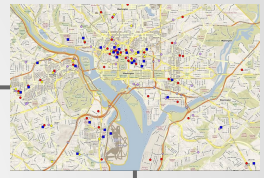
- ➔ Using Spatial Data
- ➔ GeoMesa: Scaling up
- ➔ **Spatial Analytics**

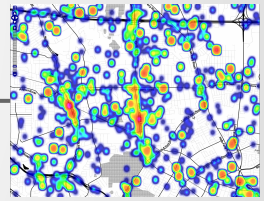
Spatial Analytic Pipeline

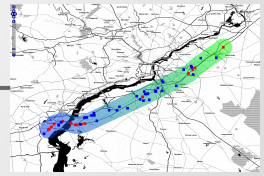


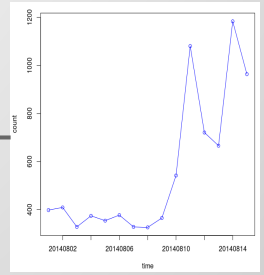
Spatial Analytic Pipelines

storage | spatial engine/api | client tools | work questions

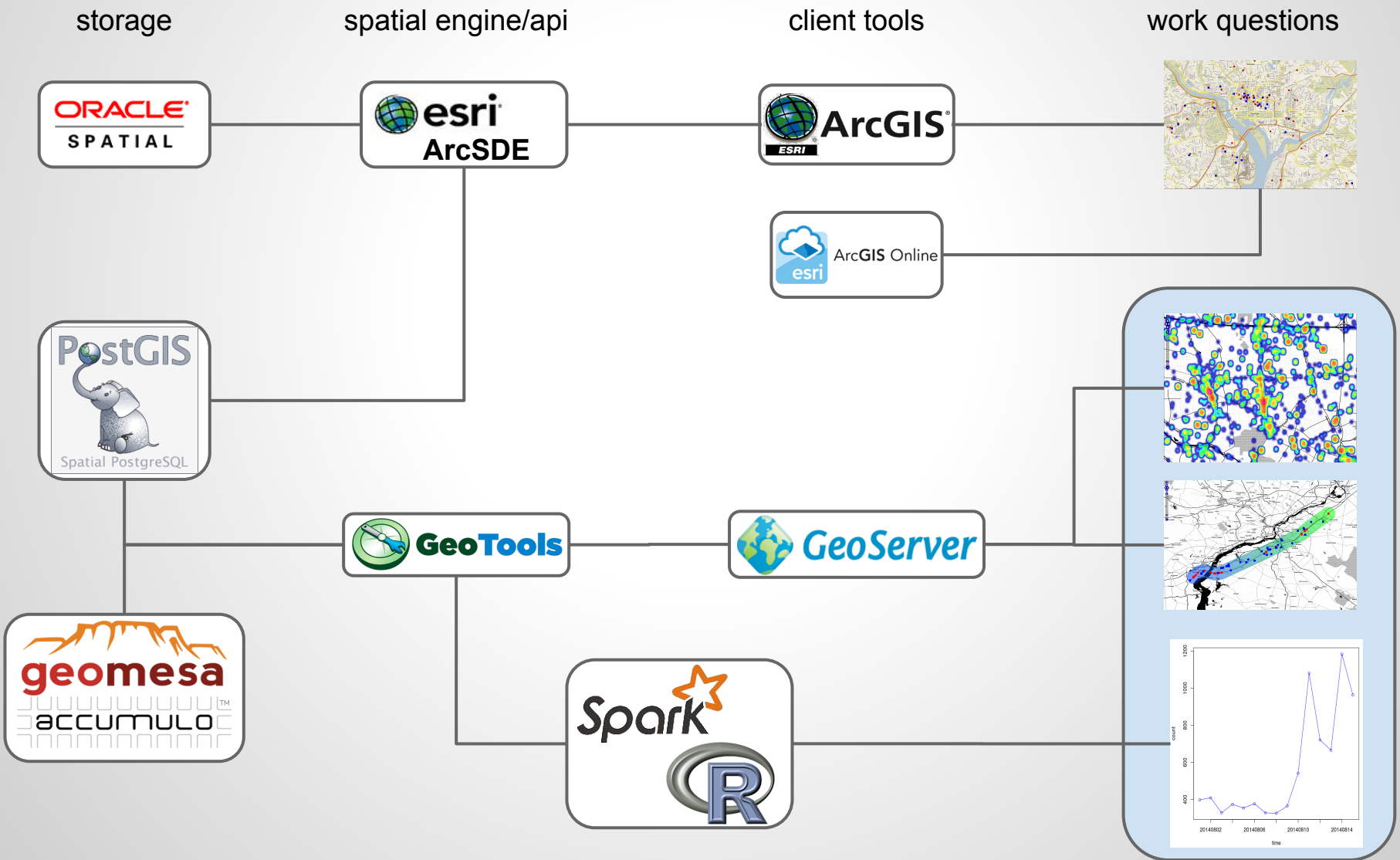




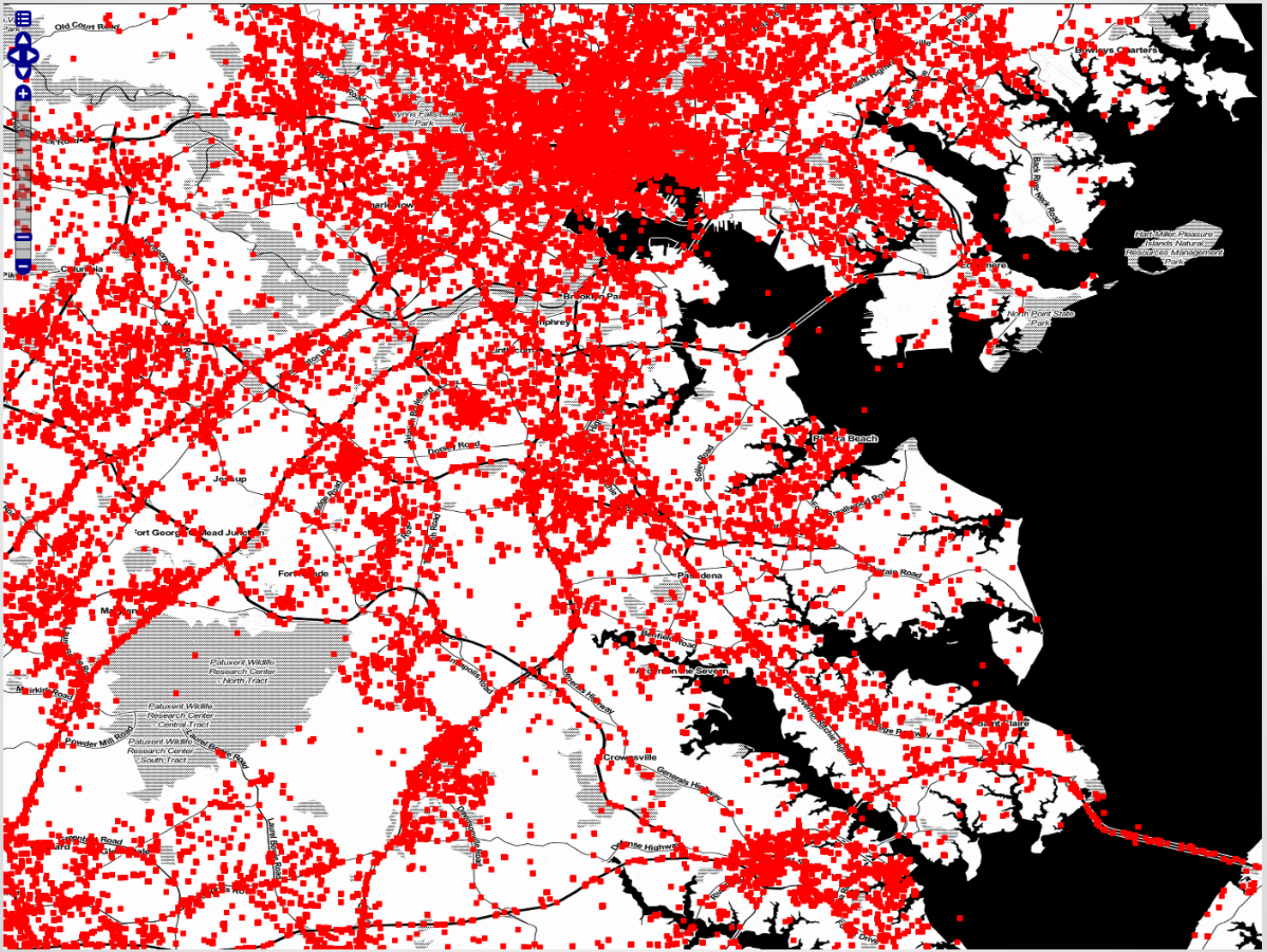




Spatial Analytic Pipelines

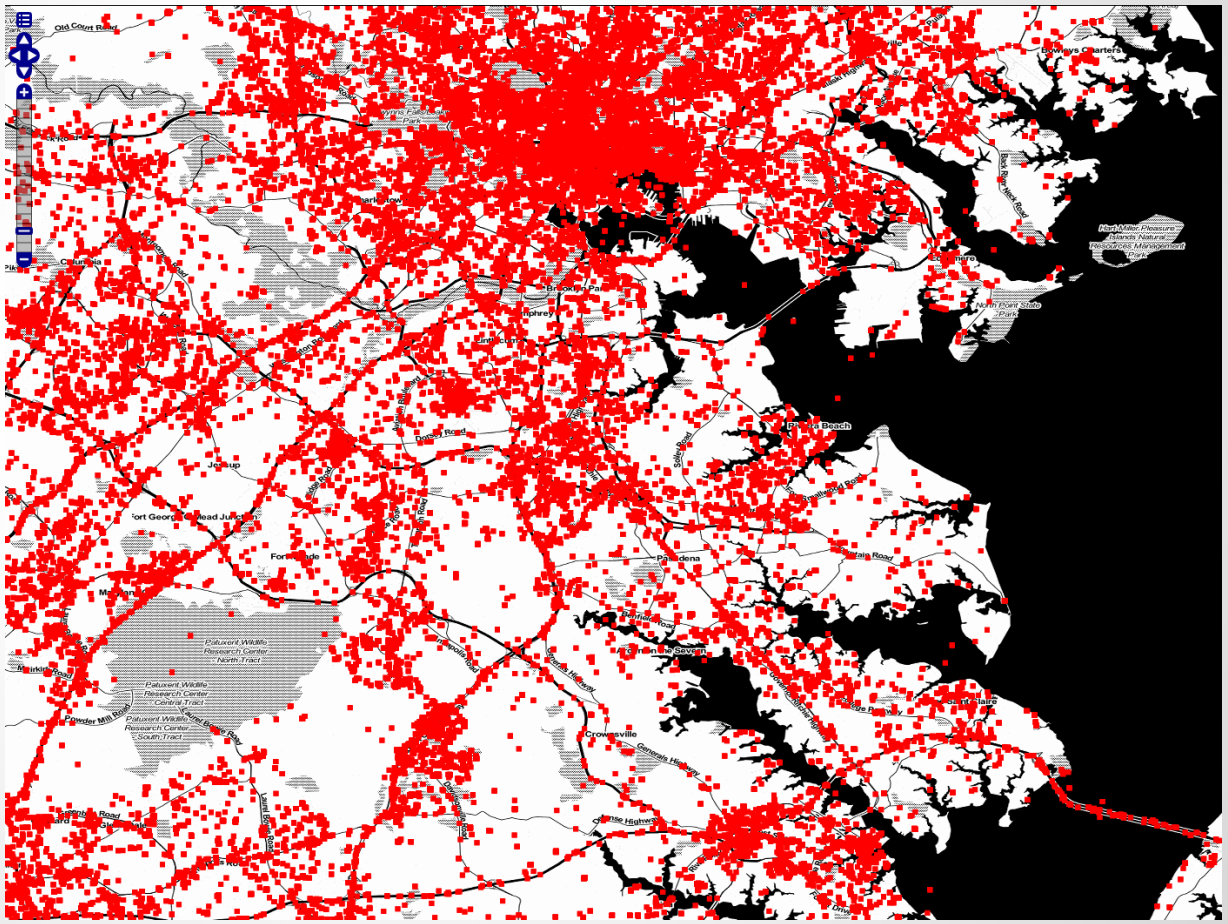


Example 1: Crowd Survey



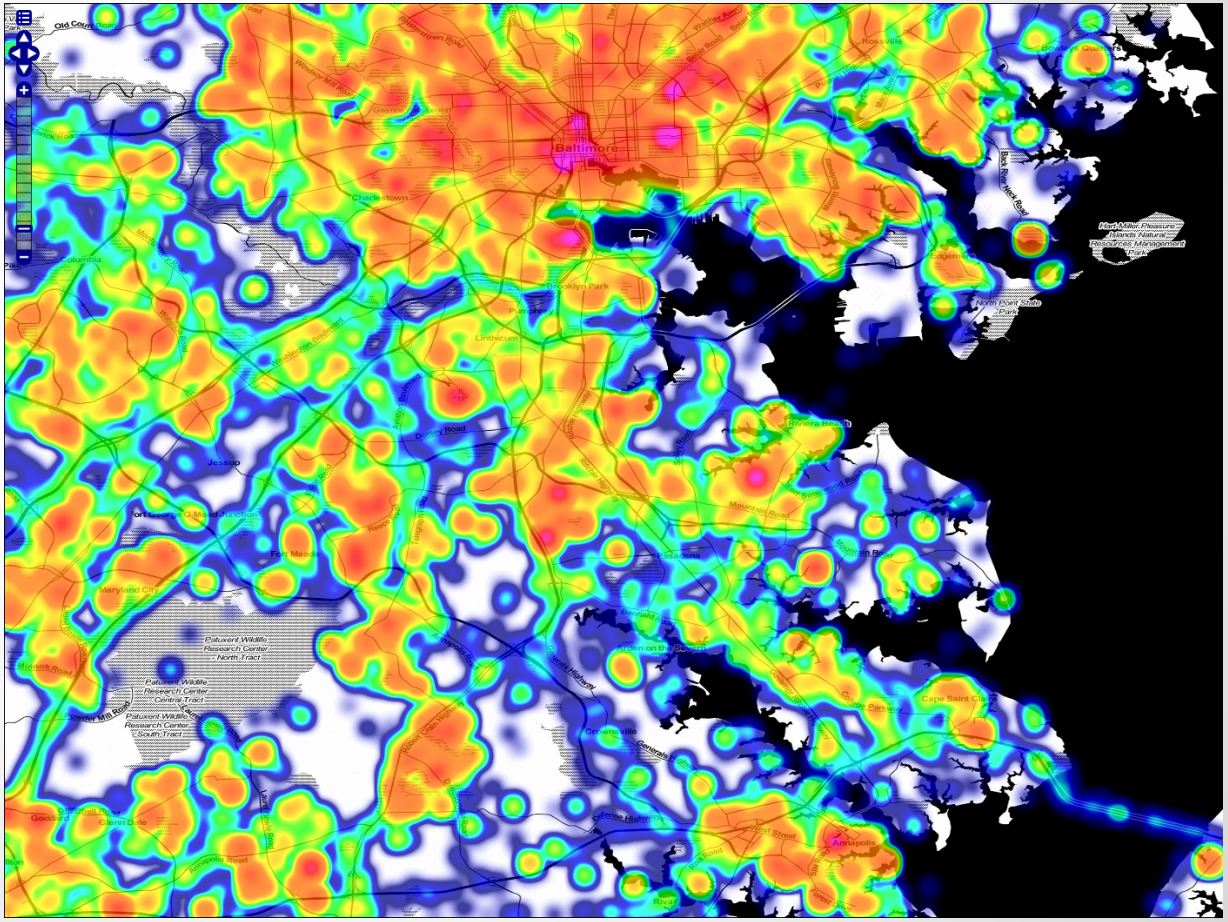
It’s...Tweeting (while Driving?)

Lets go
from this...



It’s...Tweeting (while Driving?)

to this...



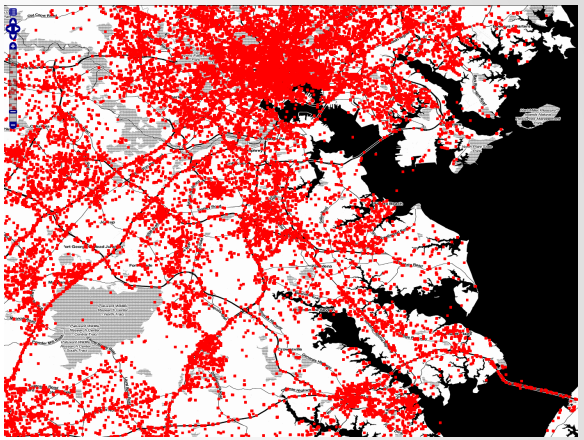
Thoughts on Density…and Scale

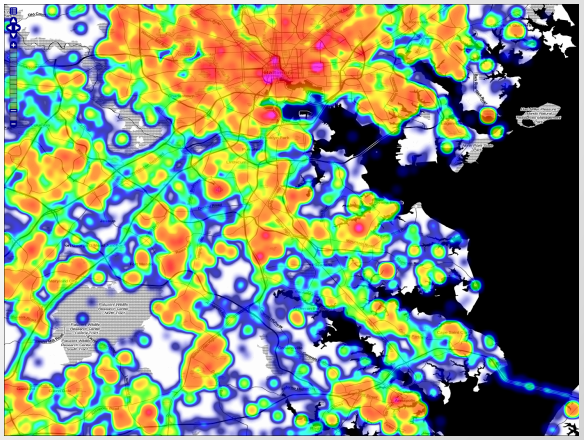
Easy things
1,000
10,000

Harder things
1,000,000
100,000,000

Very hard things
10 billion
100 billion
trillions

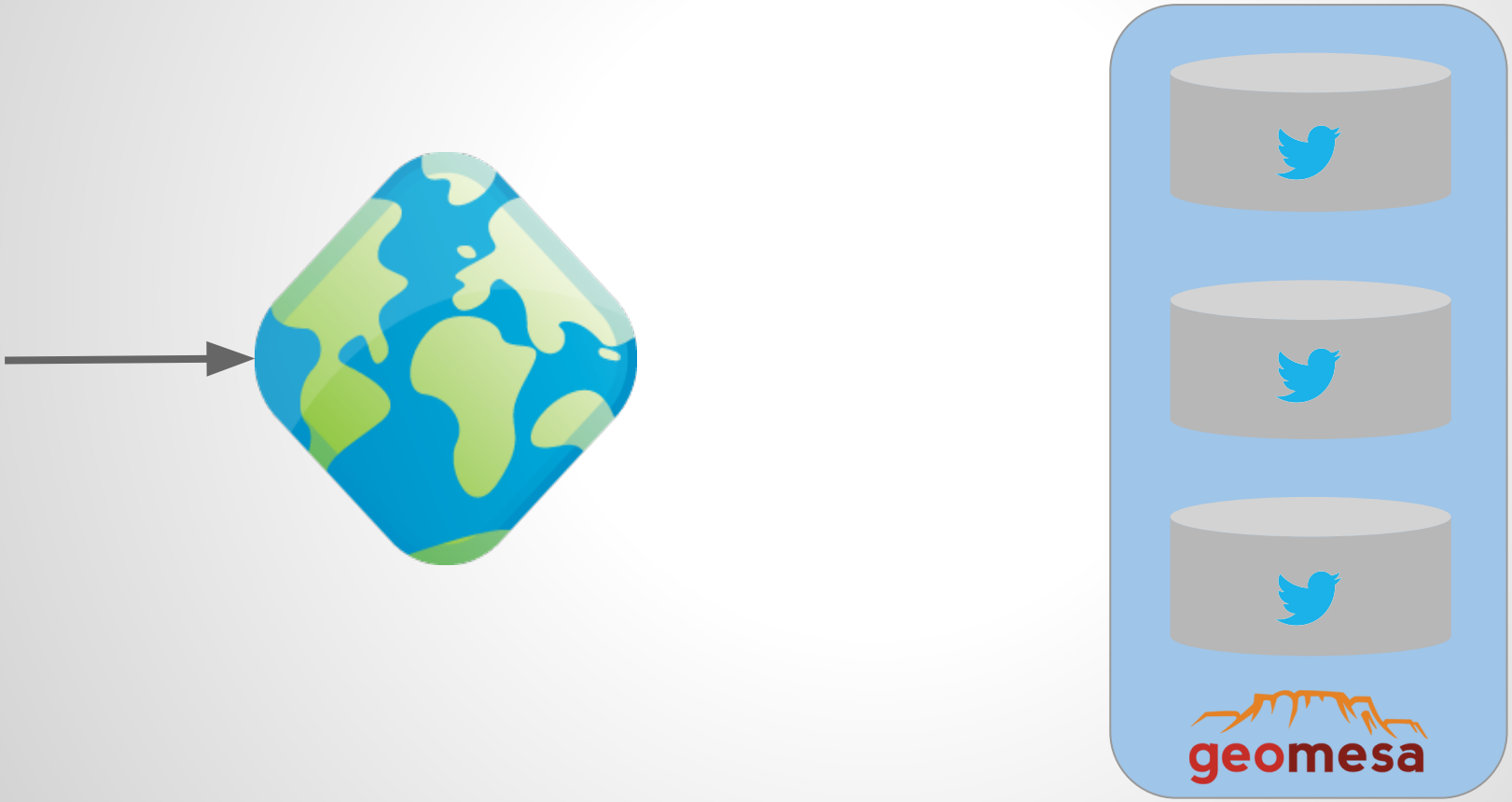
Scale
10k = 0.001 % of 1b
1s -> 100,000s (27hr)



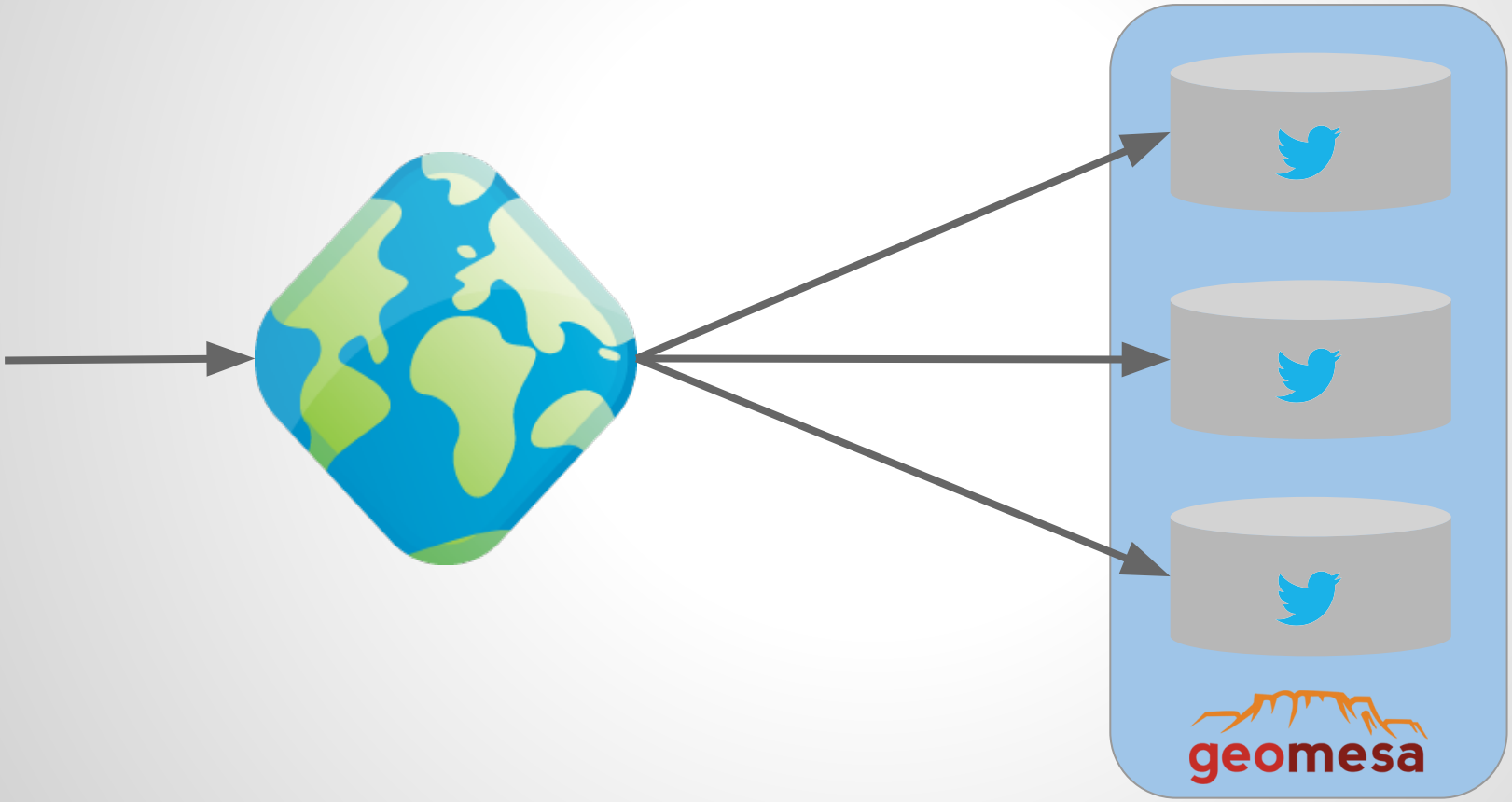


Achieving Density Parallelism

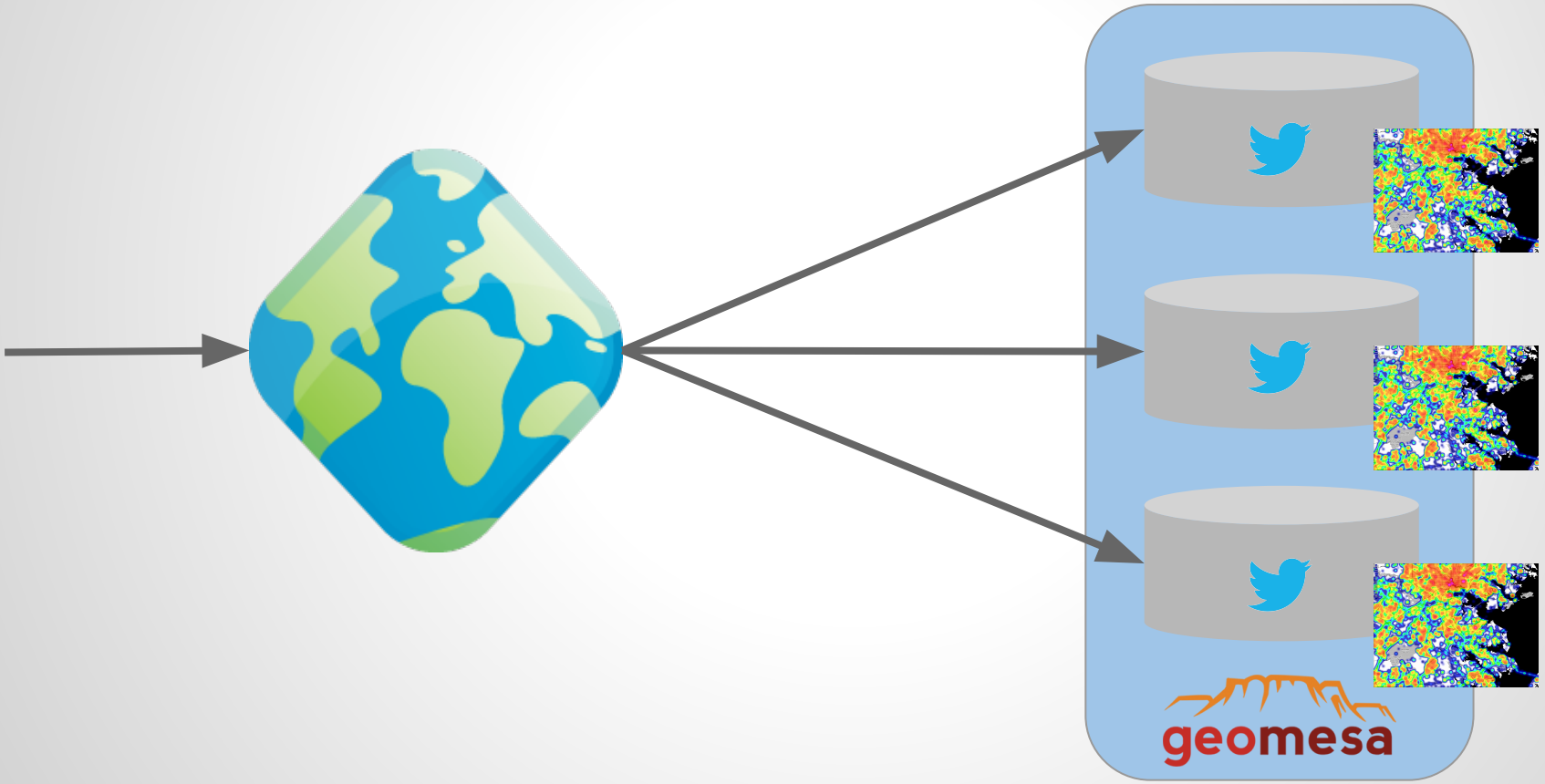
Achieving Density Parallelism



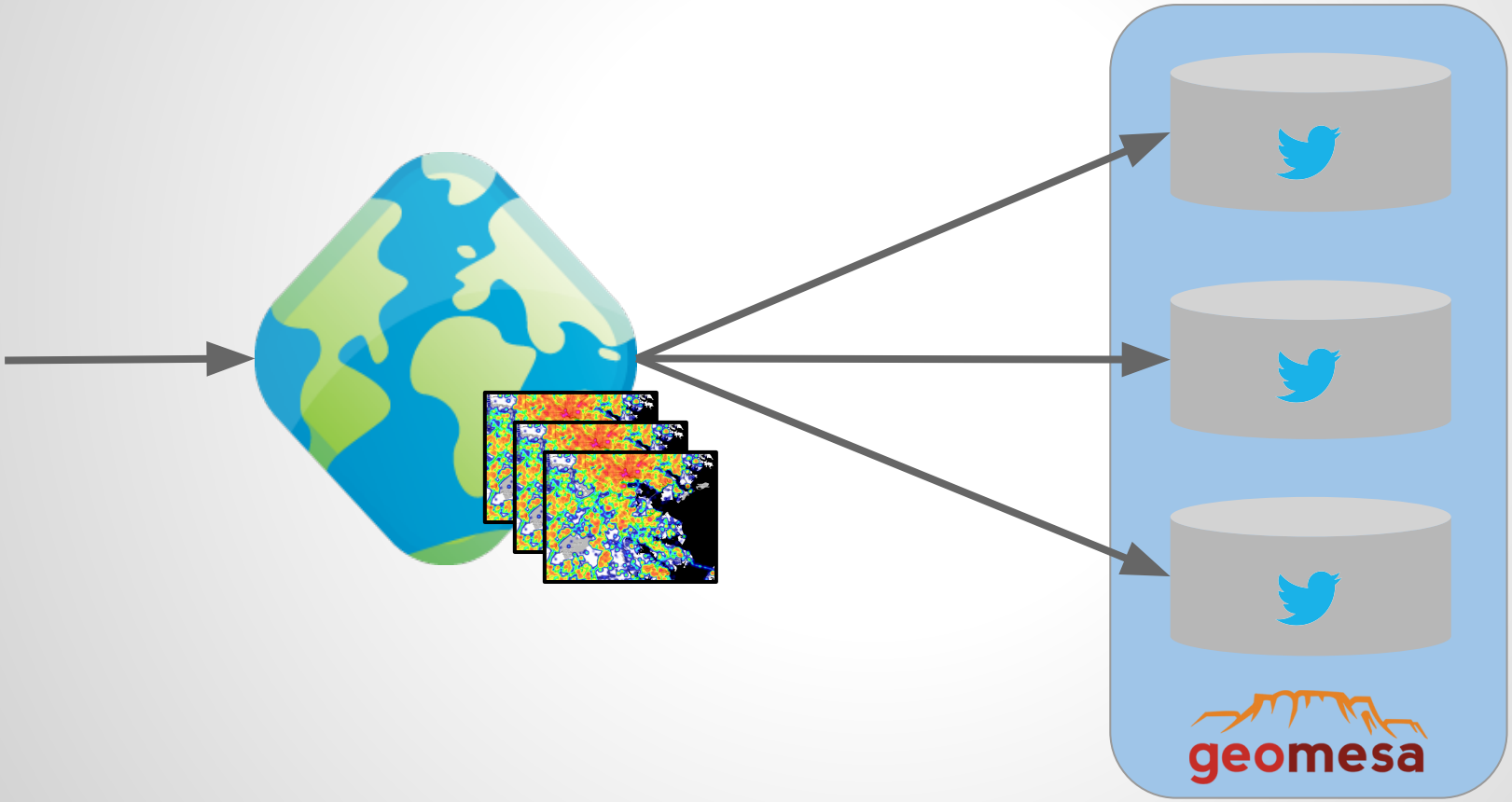
Achieving Density Parallelism



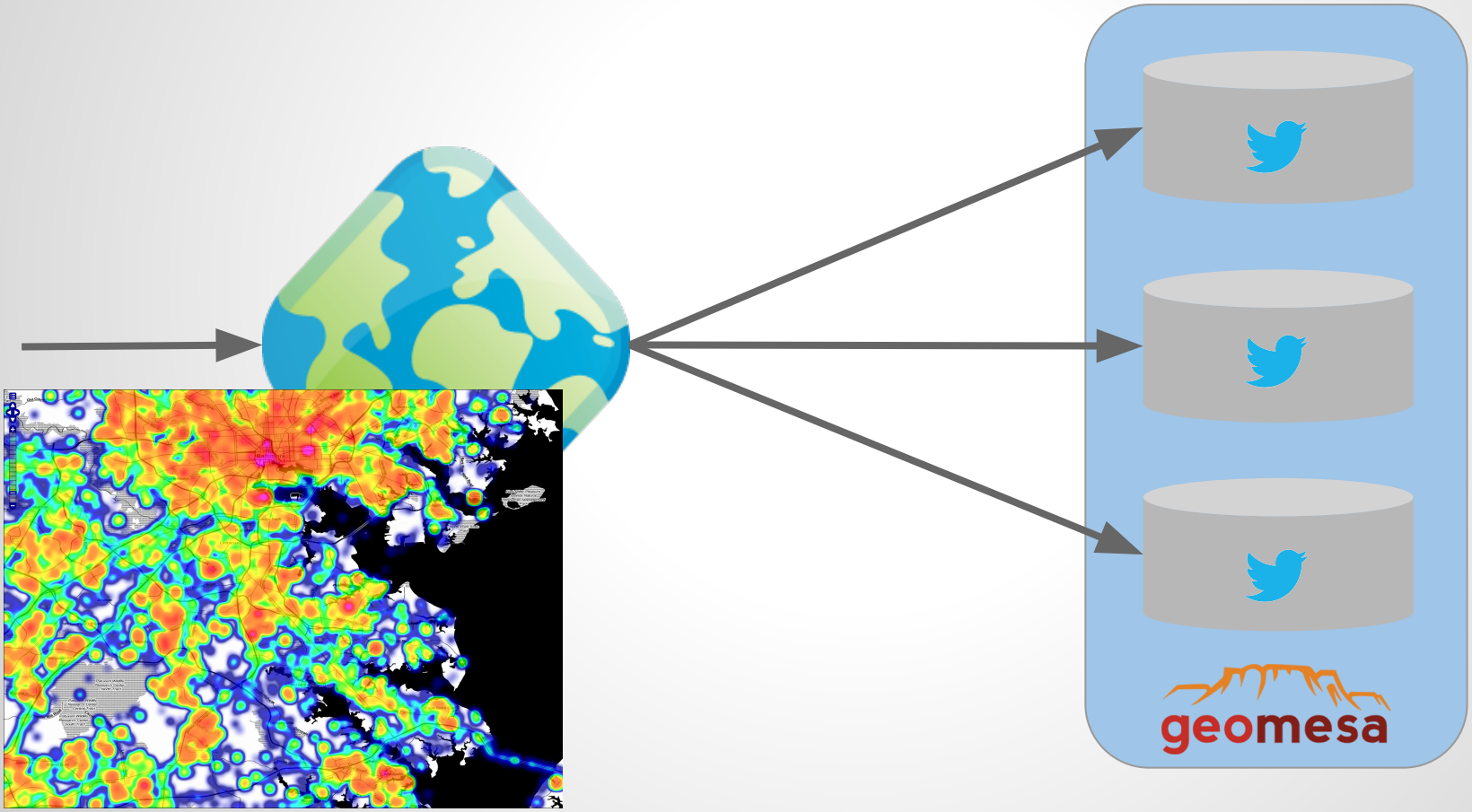
Achieving Density Parallelism



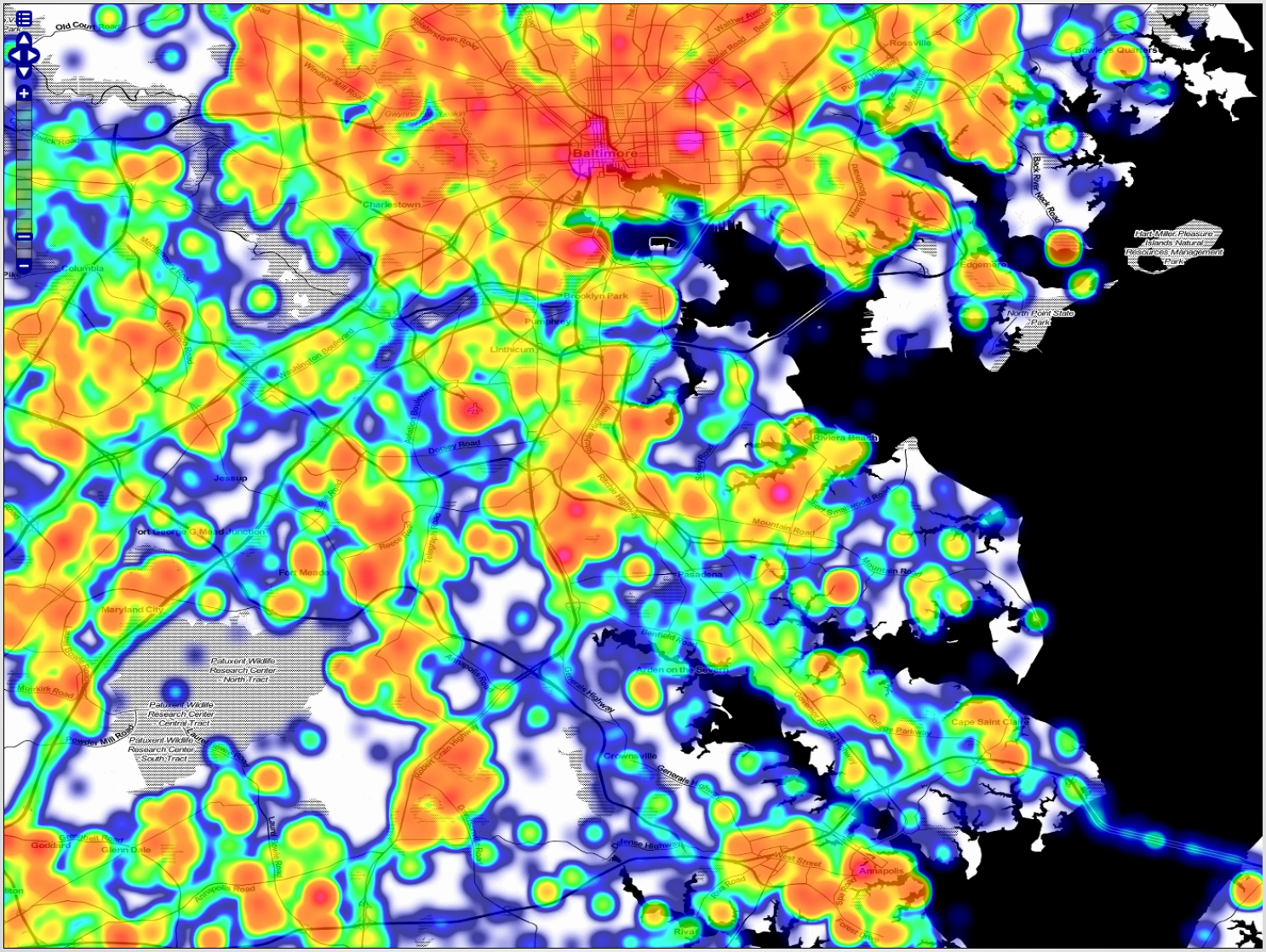
Achieving Density Parallelism



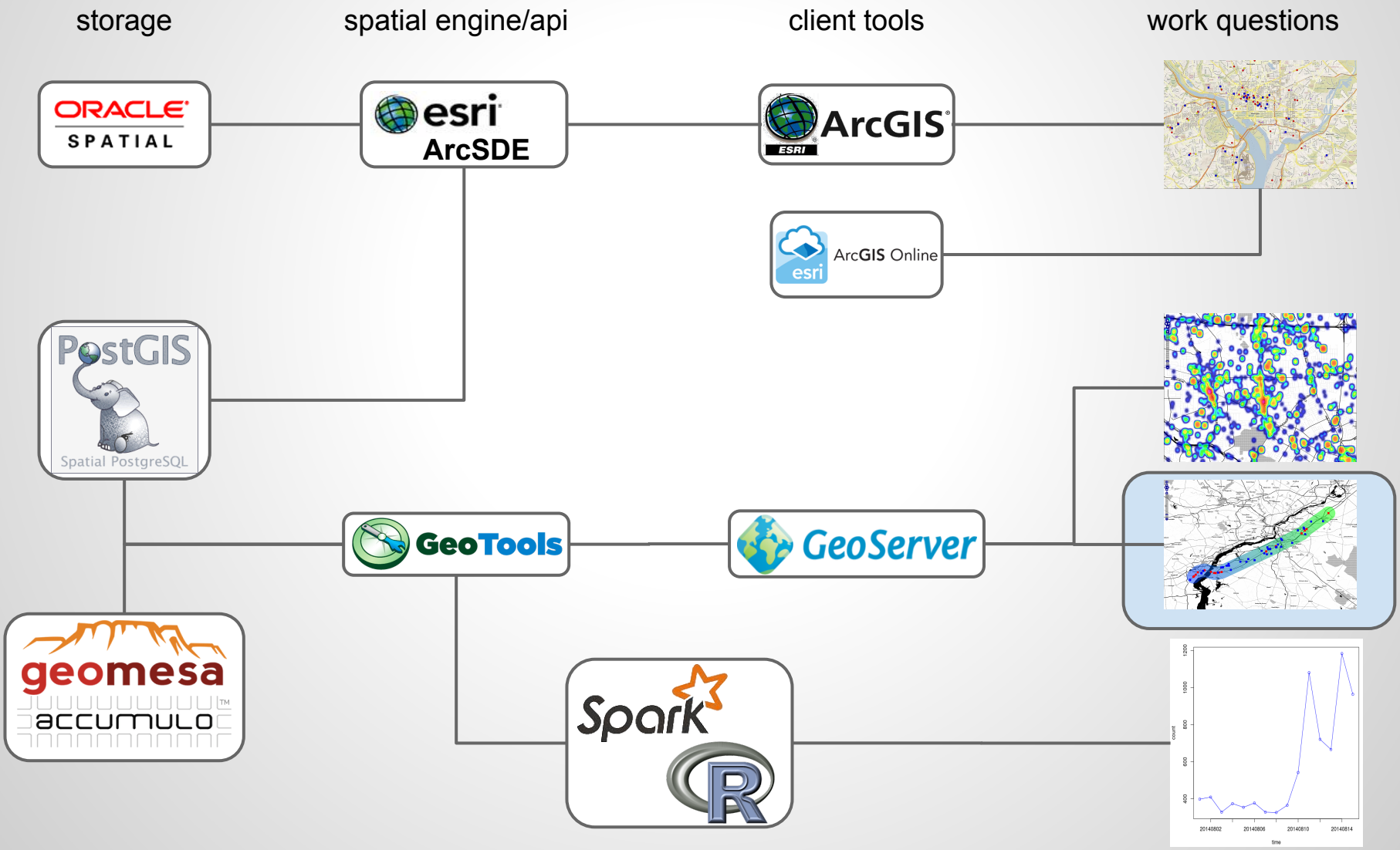
Achieving Density Parallelism



Achieving Density Parallelism



Spatial Analytic Pipelines



Ex 2: Time Interpolated Queries

1. events that were within **10 miles of my route** as I drove around Philadelphia

Ex 2: Time Interpolated Queries

1. events that were within **10 miles of my route** as I drove around Philadelphia

2. events that were within **10 miles of my route** as I drove around Philadelphia on **June 19**

Ex 2: Time Interpolated Queries

1. events that were within **10 miles of my route** as I drove around Philadelphia

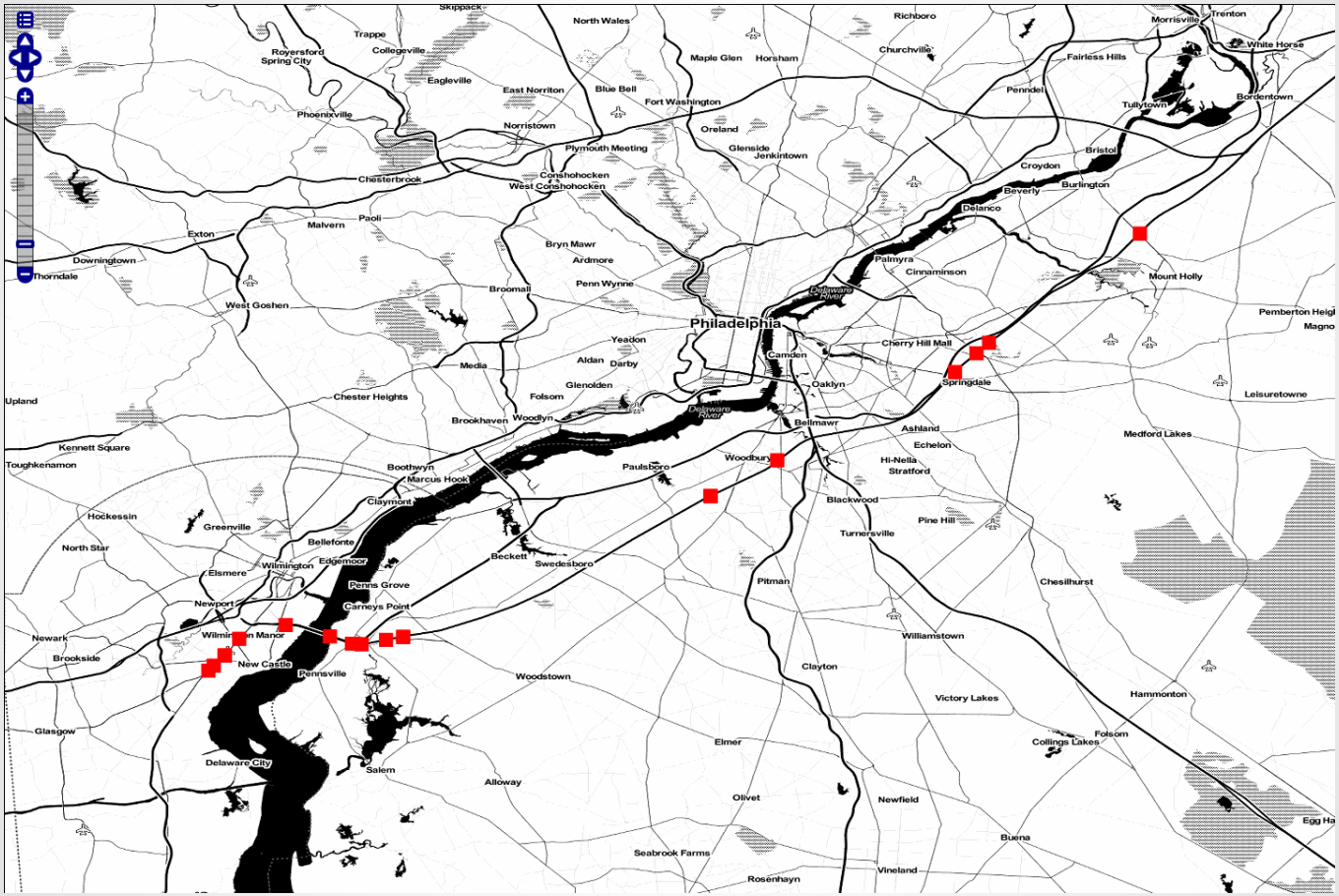
2. events that were within **10 miles of my route** as I drove around Philadelphia on **June 19**

3. events that were within **10 miles of my route** AND within **15 minutes of me** as I drove around Philadelphia

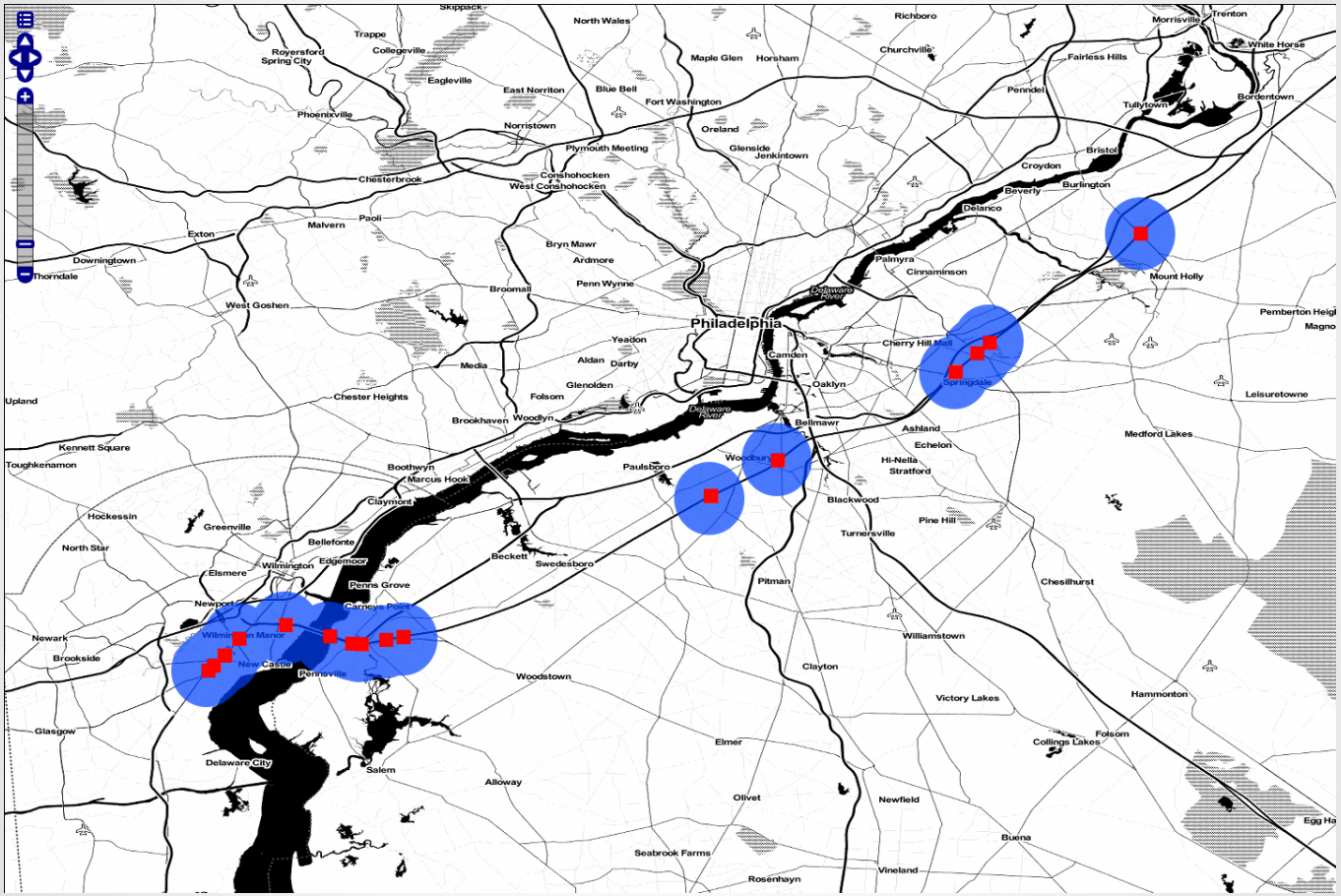
Quick Pause for WPS

- WPS - Web Processing Services
  - OGC analytic process framework
- Implement non-query processes
  - Chain analytics together
- Build your own!

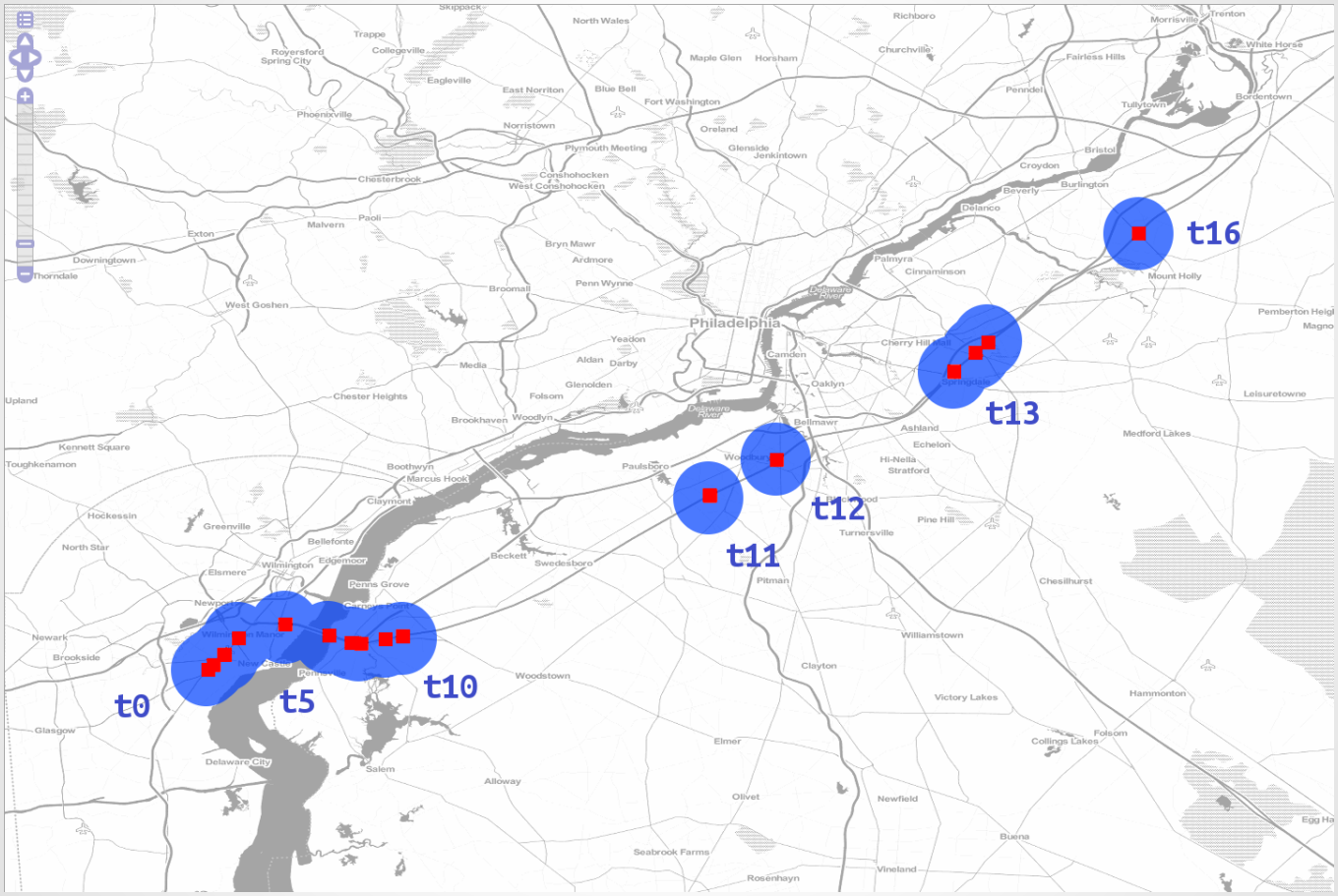
More tweeting while driving...



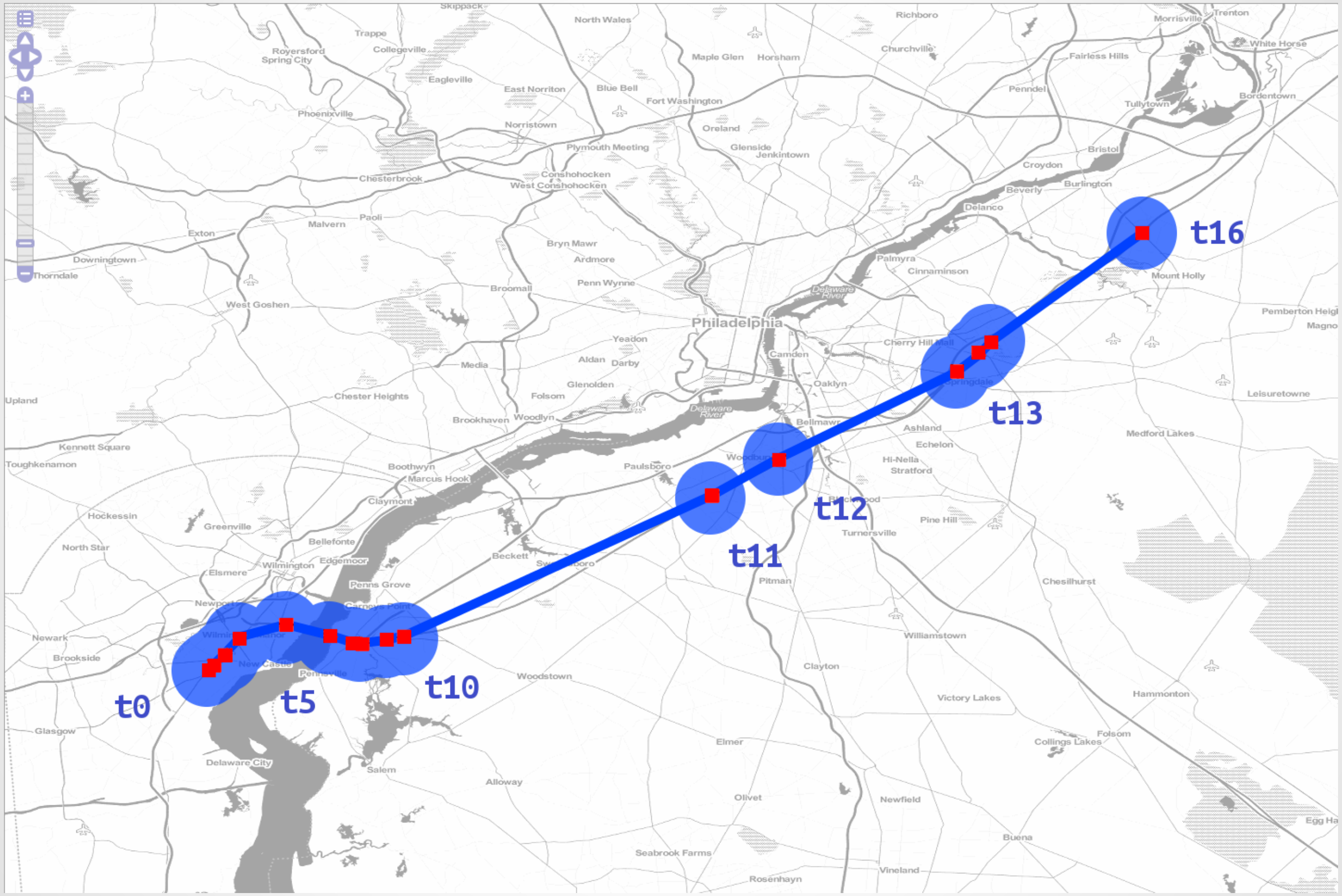
More tweeting while driving...



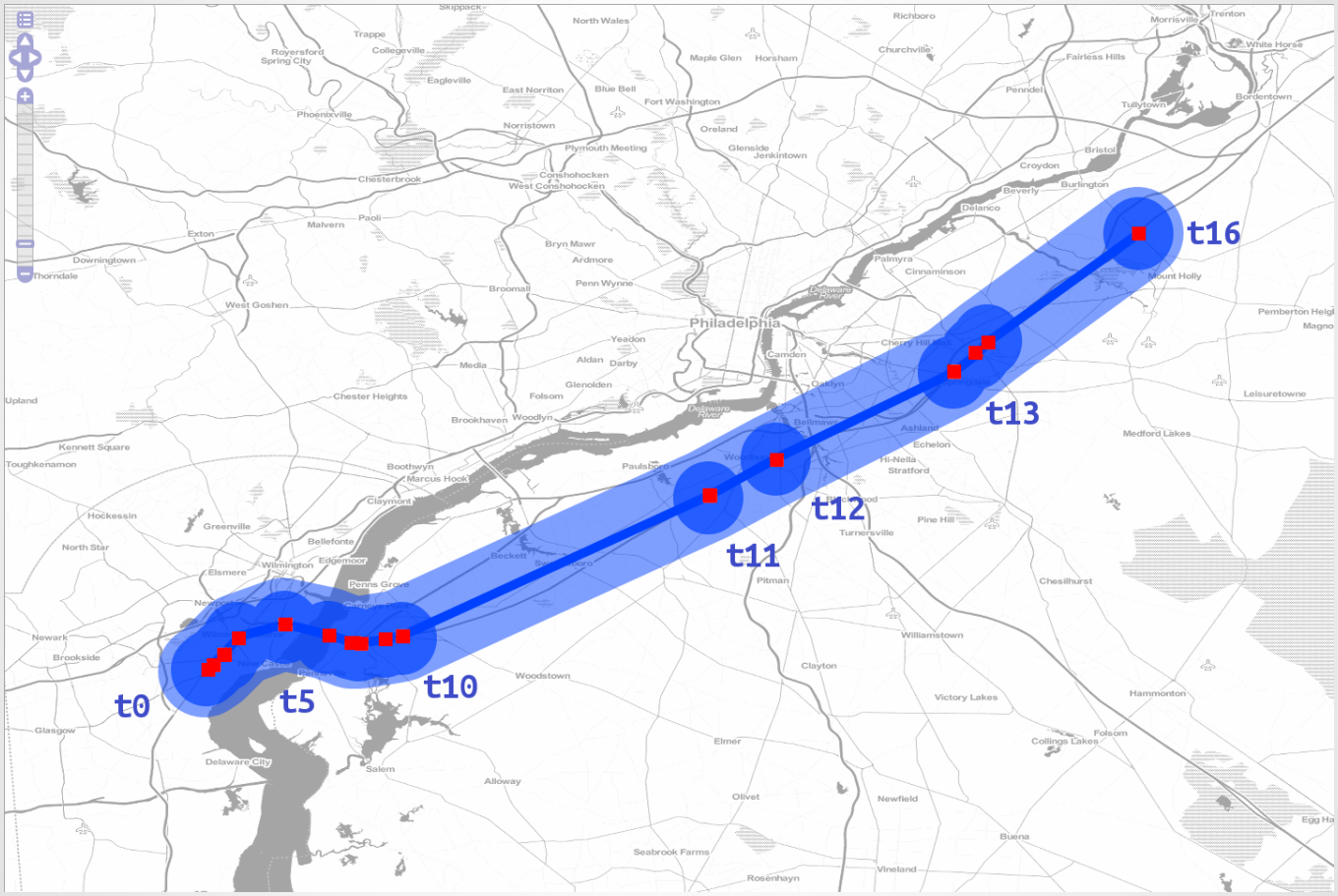
More tweeting while driving...



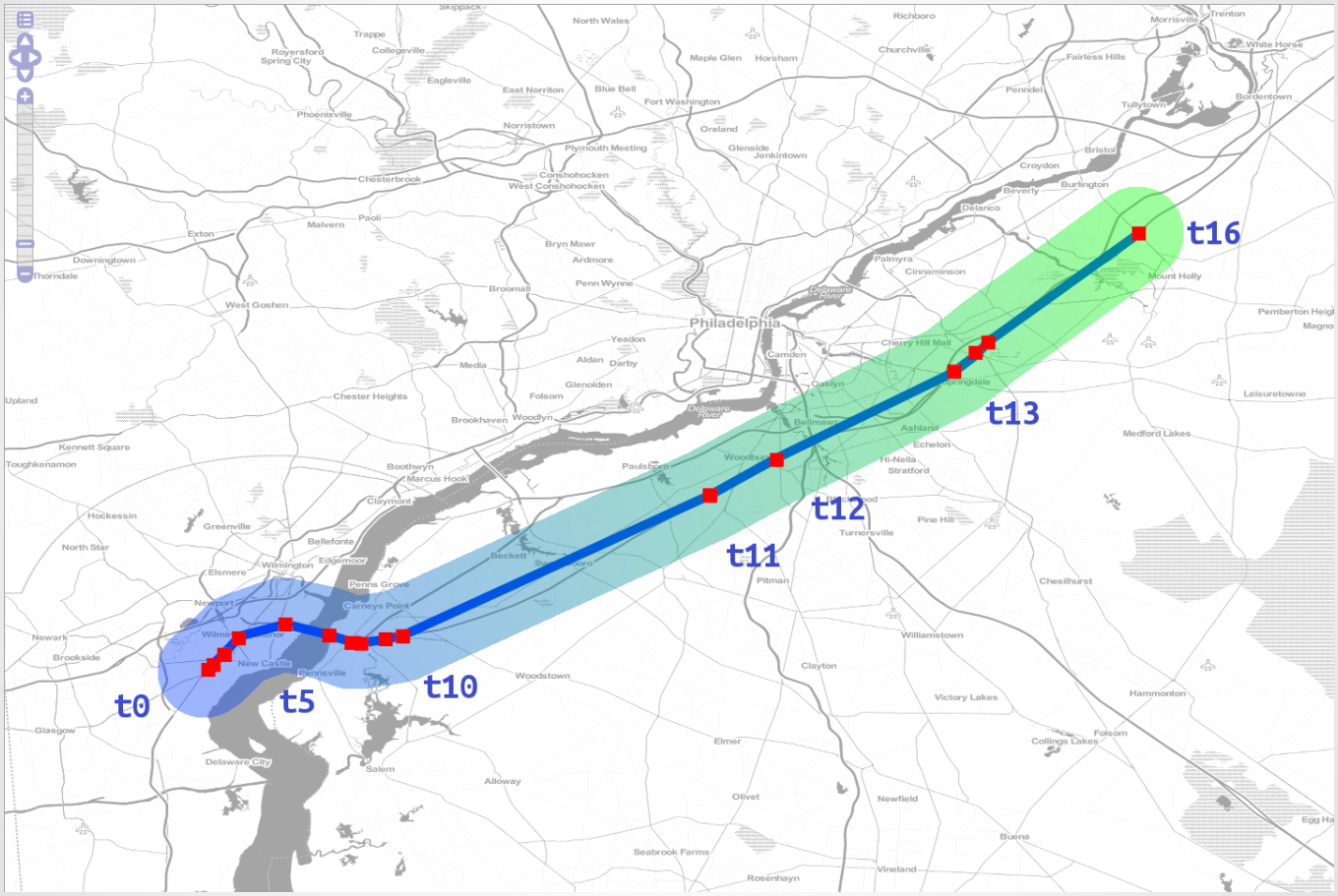
More tweeting while driving...



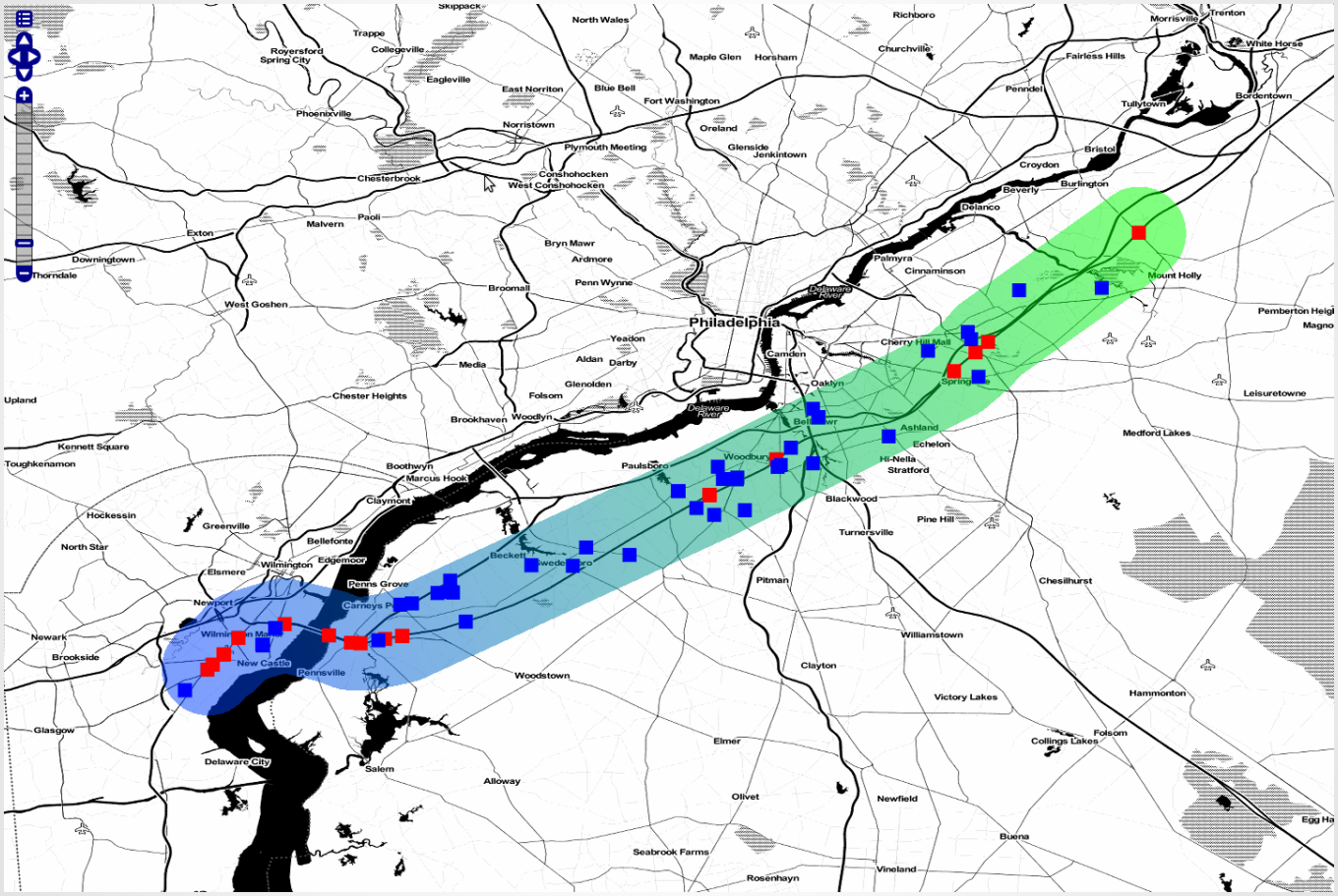
More tweeting while driving...



Interpolated Query Plan

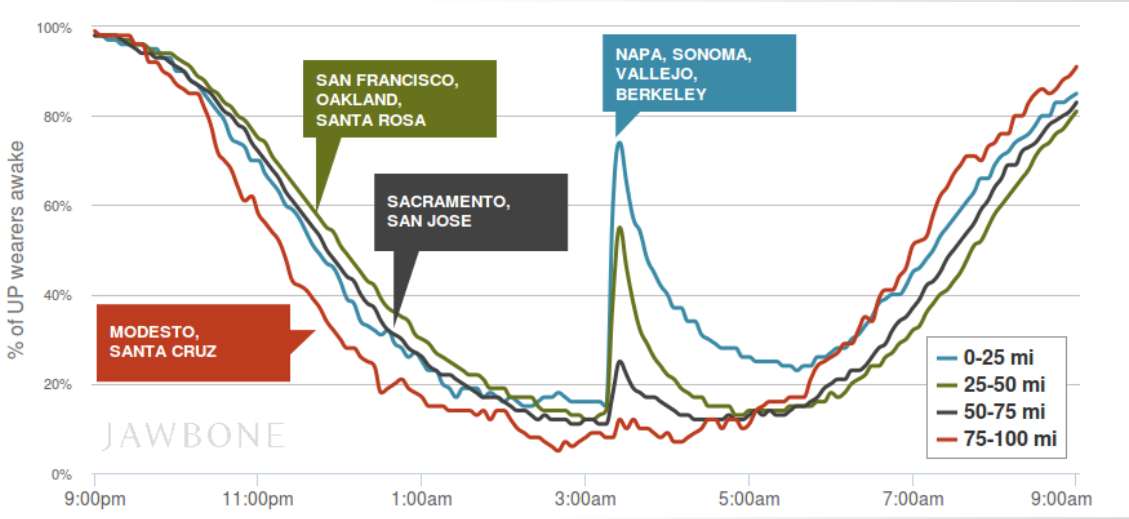


Near in space and time



Ex 3: Streaming Anomaly Detection

- Epidemiology
- Geofencing
- Tracking
- Event detection

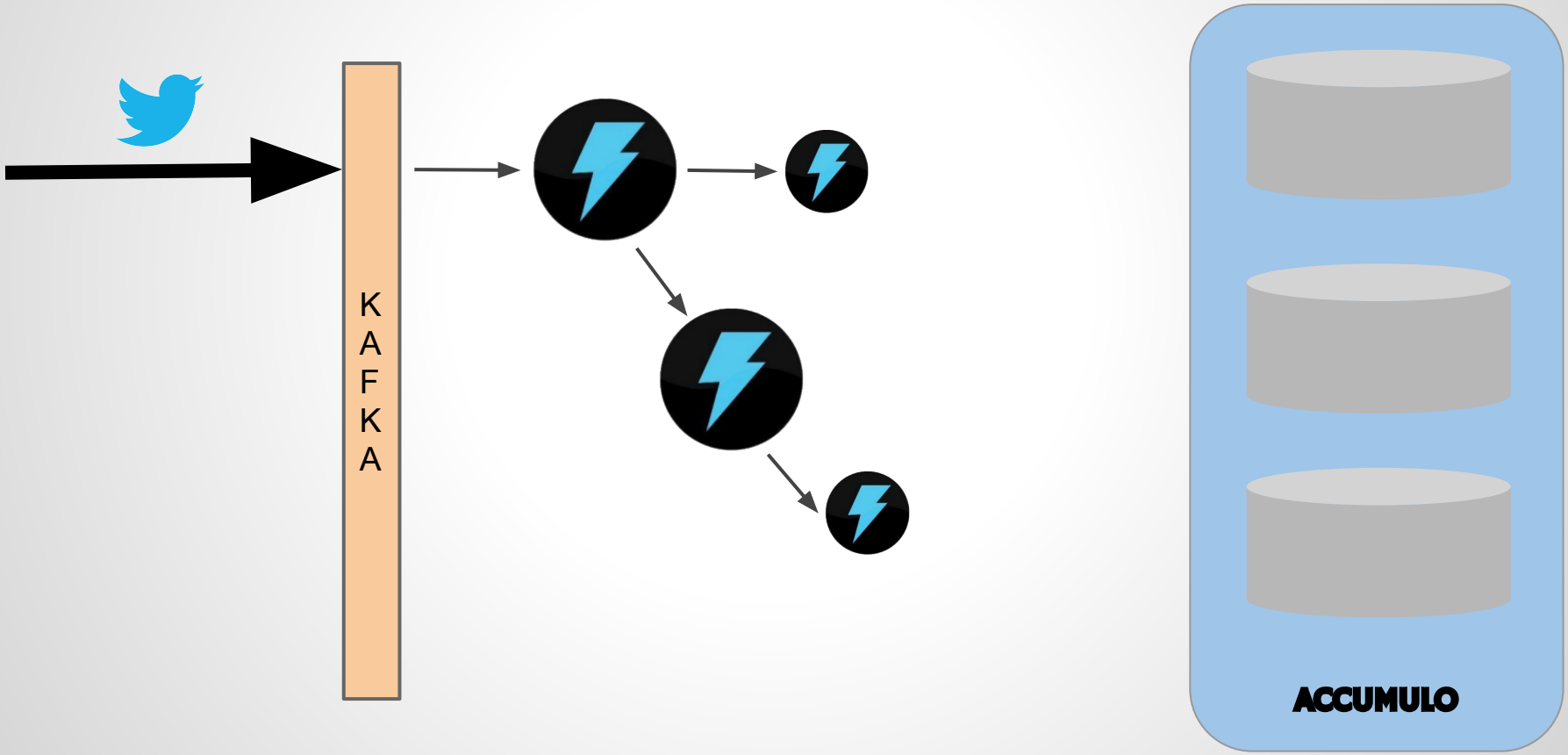


Example: Monitor Twitter stream for mentions of earthquakes. Infer sleep patterns of people in affected areas. Cluster mentions to resolve likely epicenter neighborhood. (The closer people are…the higher percentage are awake)

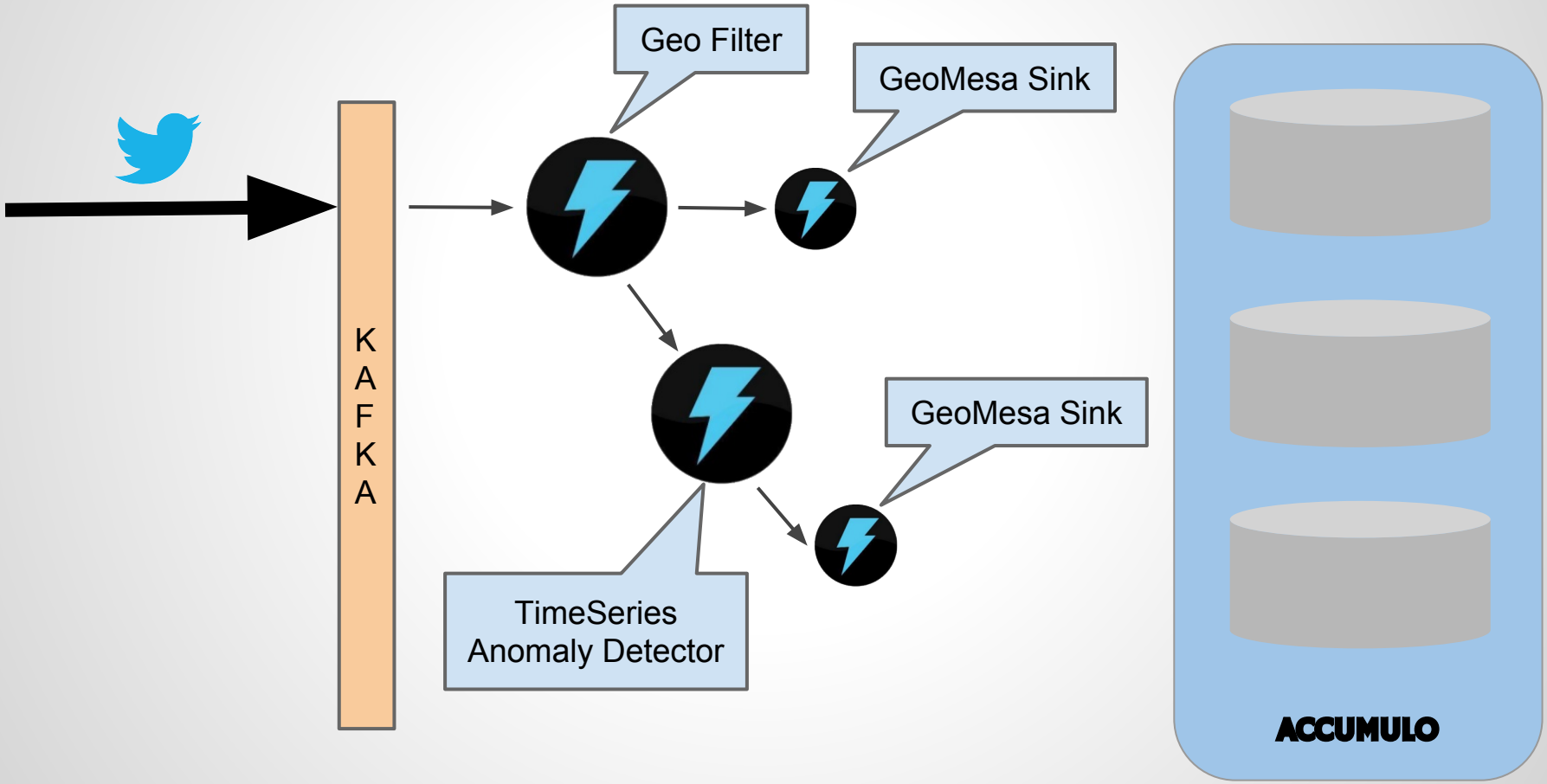
Source: https://jawbone.com/blog/napa-earthquake-effect-on-sleep/

Streaming Anomaly Detection

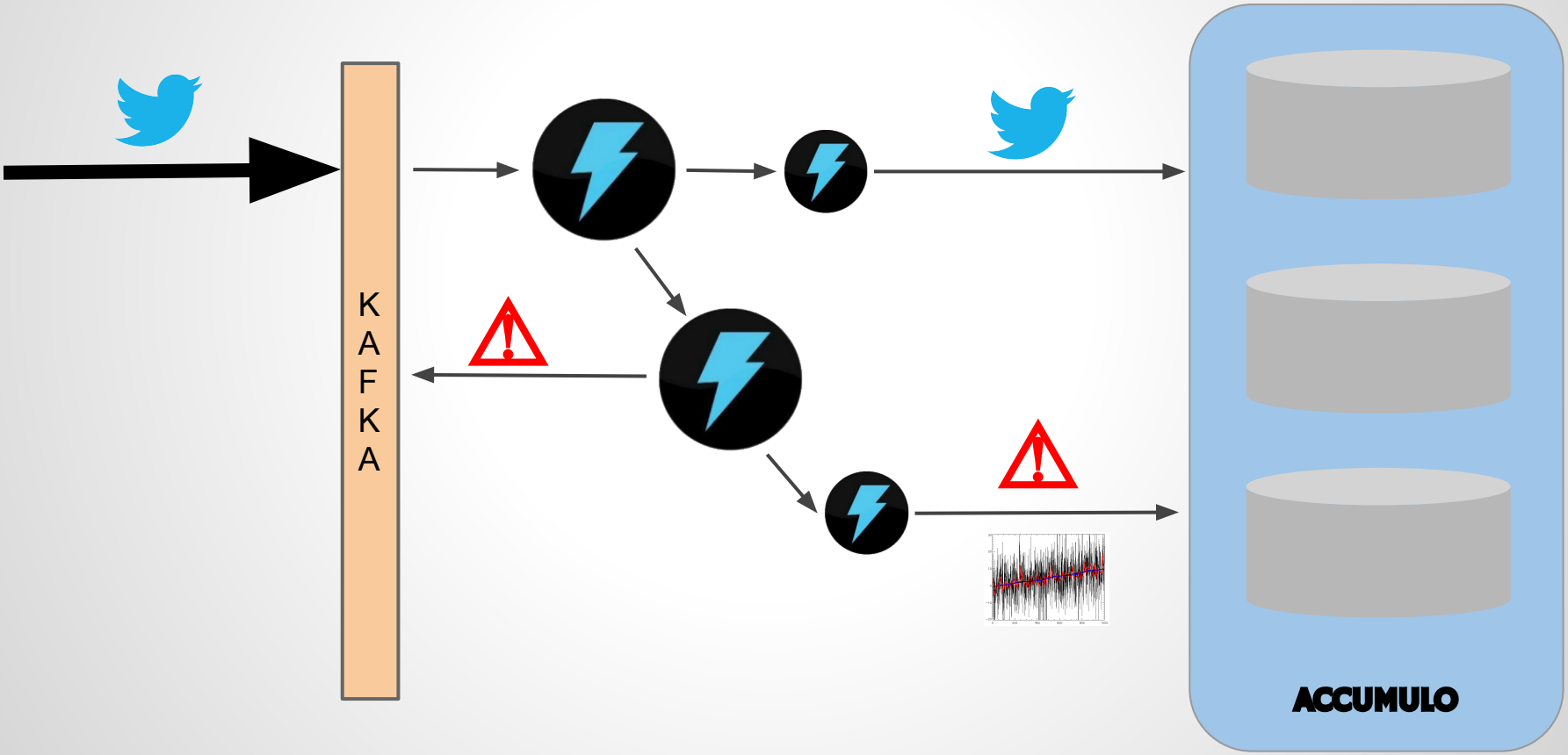
Streaming Anomaly Detection



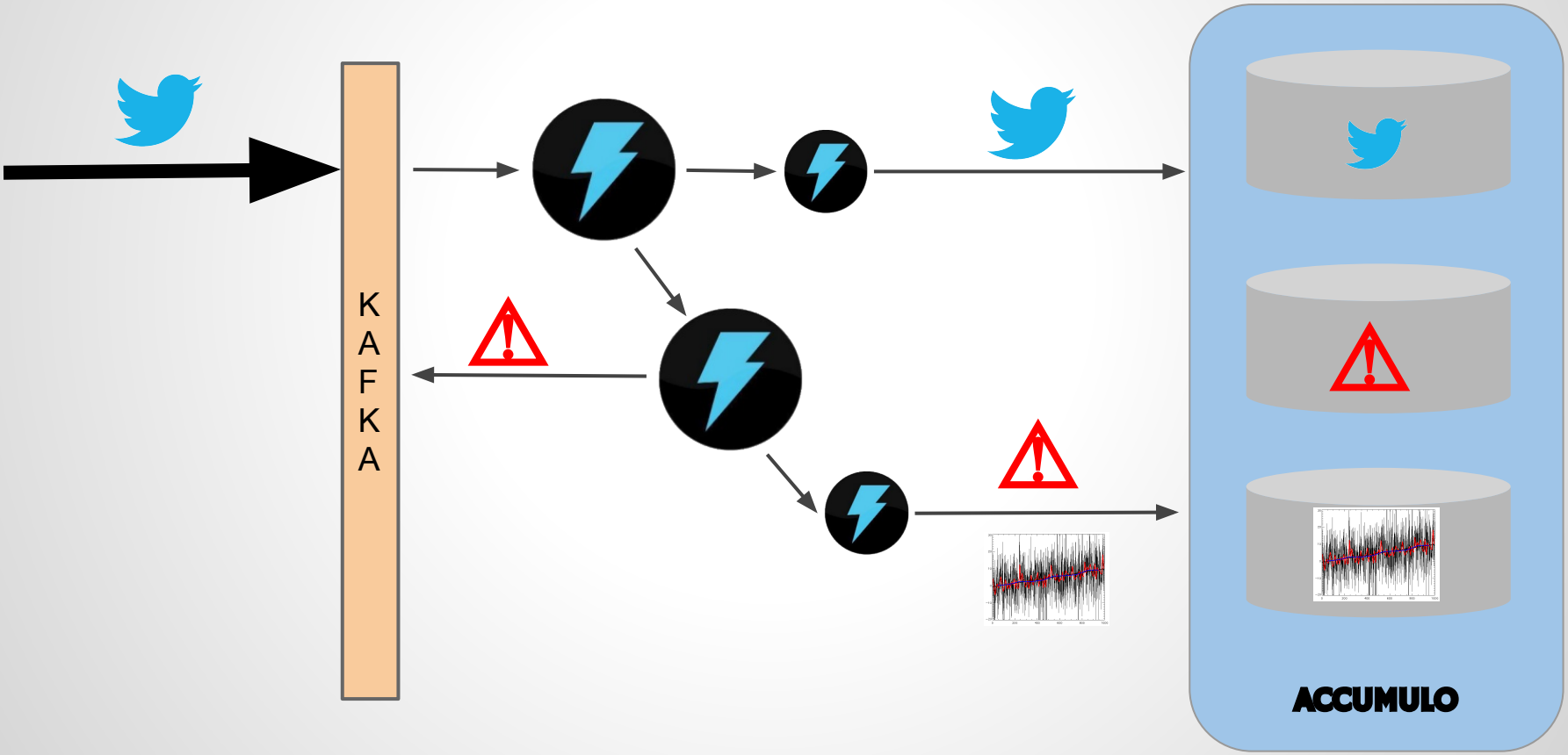
Streaming Anomaly Detection



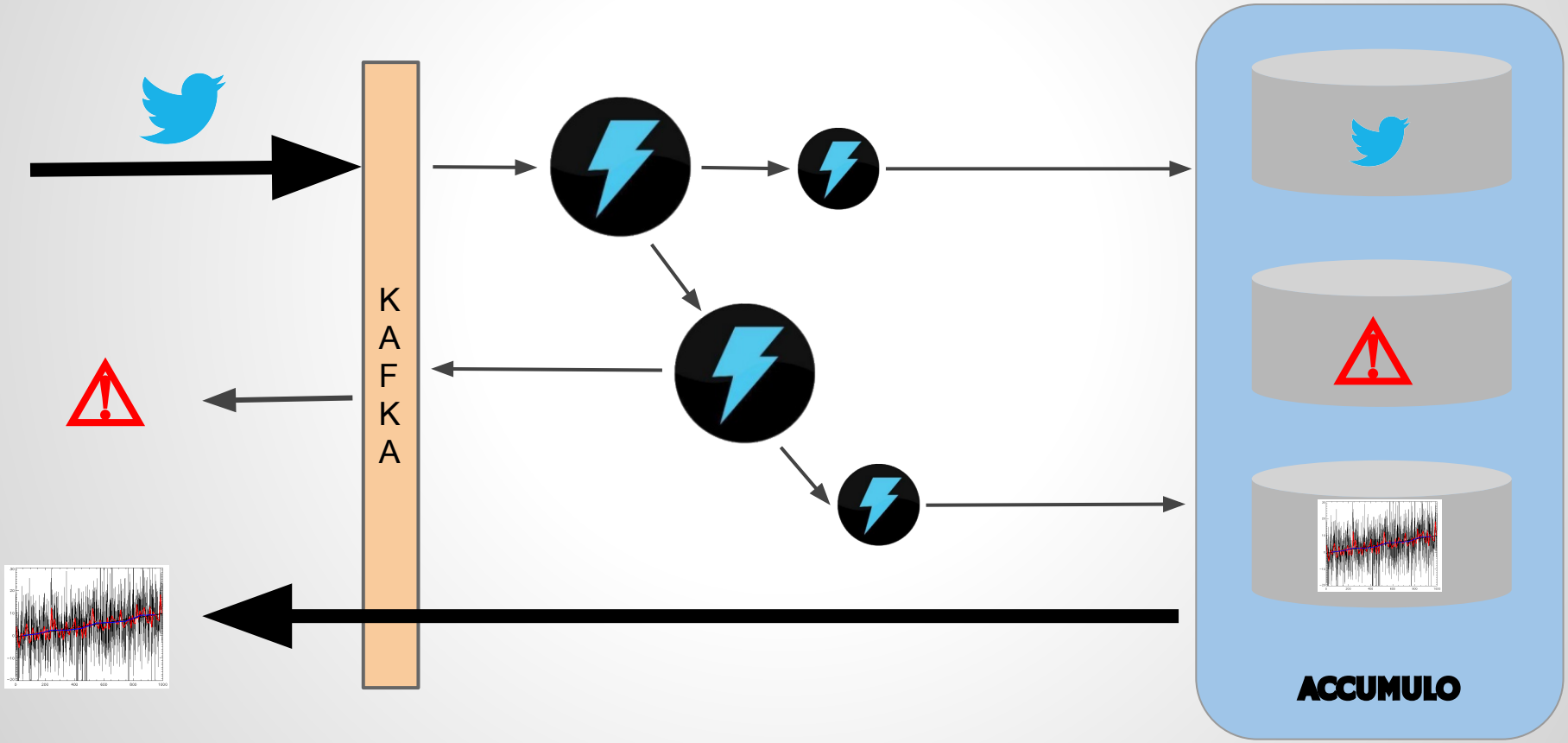
Streaming Anomaly Detection



Streaming Anomaly Detection



Streaming Anomaly Detection



Time Series Analysis

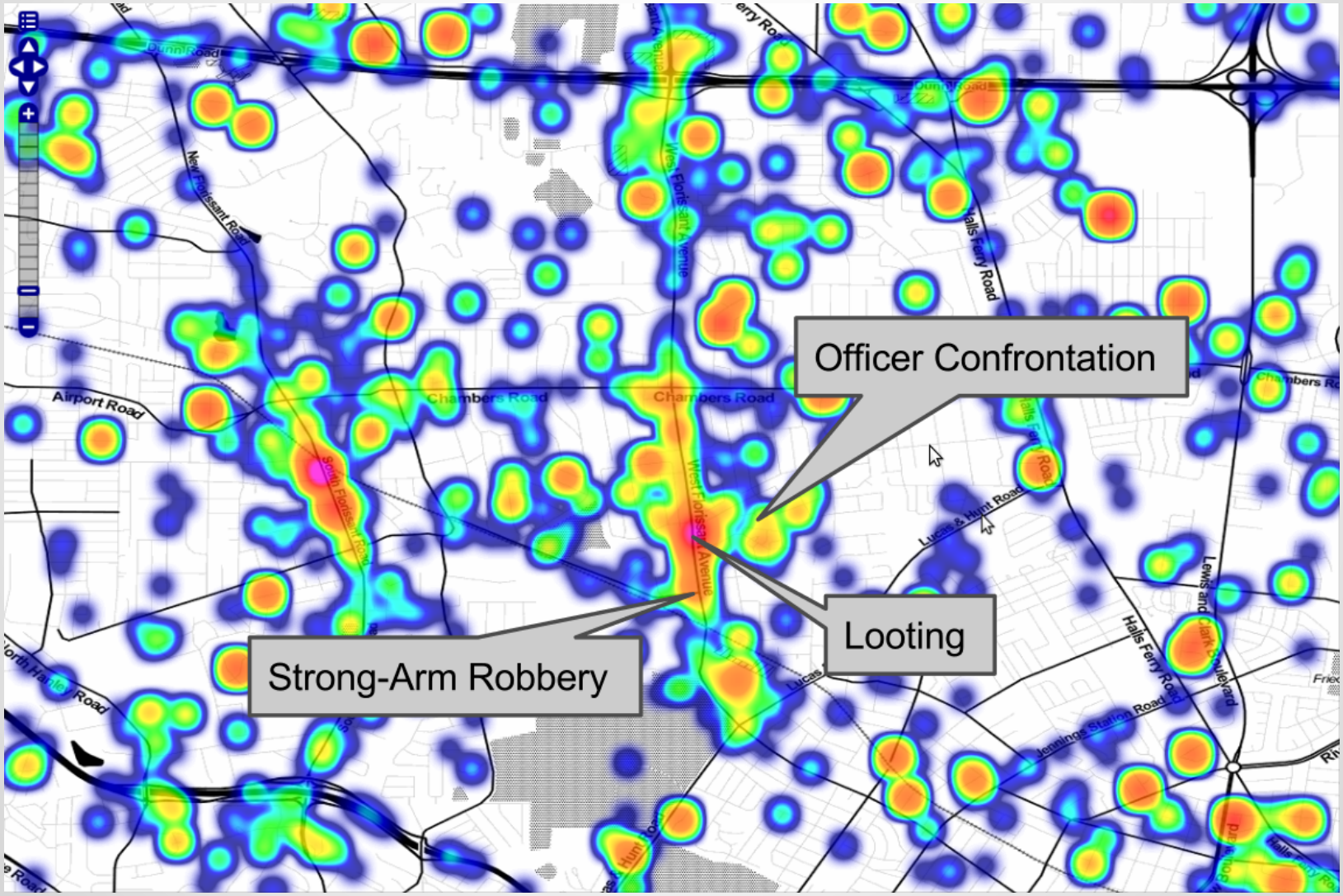
Challenge: Lift Data into Existing Tool (R) for analysis

Challenge: Massive data for traditional analysis

Enter: Apache Spark

- Real-time Data Analysis in Memory
- Native transforms, filters, aggregation

Events in Ferguson



Query

```
// Get a handle to the data store
val params = Map(
  "instanceId" -> "myinstance",
  "zookeepers" -> "zoo1,zoo2,zoo3",
  "user"       -> "username",
  "password"   -> "password",
  "tableName"  -> "geomesa_catalog")

val ds = DataStoreFinder.getDataStore(params).asInstanceOf[AccumuloDataStore]

// Construct a CQL query to filter by bounding box
val ff = CommonFactoryFinder.getFilterFactory2
val f = ff.bbox("geom",  -90.32023,38.72009,-90.23957,38.77019, "EPSG:4326")
val q = new Query(feature, f)
```

Distribute

```
val conf = new Configuration
val sconf = init(new SparkConf(true), ds)
val sc = new SparkContext(sconf)

val queryRDD = geomesa.compute.spark.GeoMesaSpark.rdd(conf, sconf, ds, query)
```

Aggregate

```
// Convert RDD[SimpleFeature] to RDD[(String, SimpleFeature)] where the first
// element of the tuple is the date to the day resolution
val dayAndFeature = queryRDD.mapPartitions { iter =>
  val df = new SimpleDateFormat("yyyyMMdd")
  val ff = CommonFactoryFinder.getFilterFactory2
  val exp = ff.property("dtg")
  iter.map { f => (df.format(exp.evaluate(f).asInstanceOf[java.util.Date]), f) }
}

// Aggregate and output
val groupedByDay = dayAndFeature.groupBy { case (date, _) => date }
val countByDay = groupedByDay.map { case (date, iter) => (date, iter.size) }
countByDay.collect.foreach(println)
```

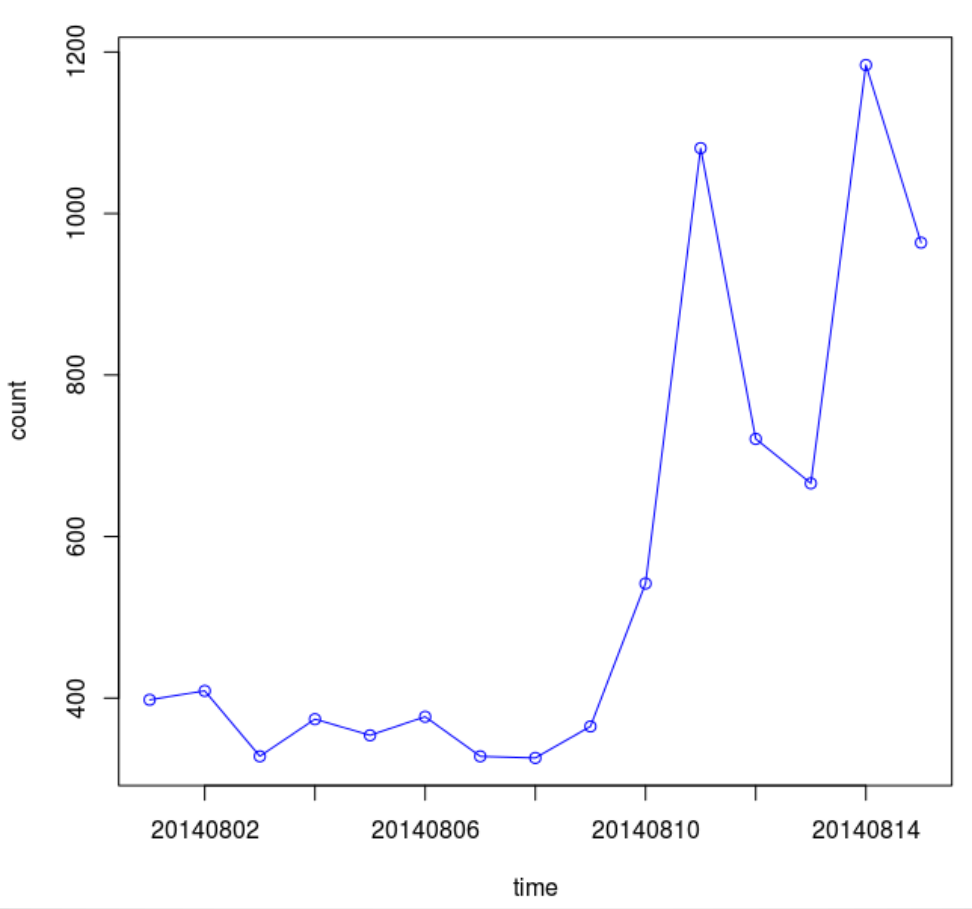
Export

```
14/08/18 01:05:57 INFO SparkContext: Job finished: collect at Runner.scala:61,
took 44.154914093 s
(20140801,398)
(20140802,409)
(20140803,328)
(20140804,374)
(20140805,354)
(20140806,377)
(20140807,328)
(20140808,326)
(20140809,365)
(20140810,542)
(20140811,1081)
(20140812,721)
(20140813,666)
(20140814,1184)
(20140815,964)
```

```
# read tweets
tweets = read.csv("~/Desktop/twitter.csv")

# plot tweets
plot(tweets,  col="blue", type="o")
```

Visualize


count
400
600
800
1000
1200
20140802
20140806
20140810
20140814
time

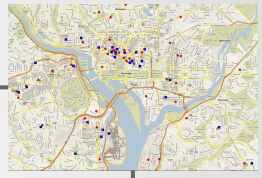
Spatial Analytic Pipelines

storage | spatial engine/api | client tools | work questions

ORACLE SPATIAL

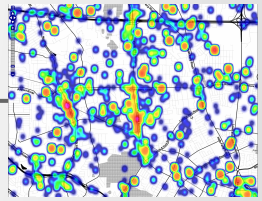
esri ArcSDE

ArcGIS ESRI



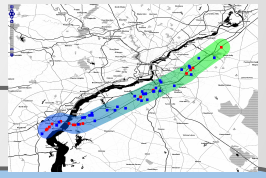
esri ArcGIS Online

PostGIS Spatial PostgreSQL



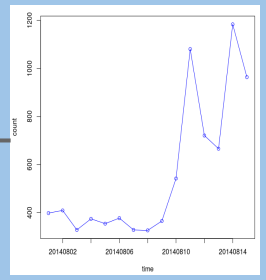
GeoTools

GeoServer



geomesa accumulo

Spark R



Other Analytics

- Spatio-temporal Event Prediction
  - Threat Surfaces to predict crime, IEDs, disease outbreaks
- DBSCAN Clustering
  - **D**ensity **B**ased **S**patial **C**lustering of **A**pplications with **N**oise
  - Clusters algorithmically - don't need to know ***k***
- KNN (K-Nearest Neighbors)
  - Find 10 Closest Bars for after the Meetup using Open Street Map
- more to come…
  - usually implemented as WPS processes

But wait! There's more! A live demo?

Source: http://en.wikipedia.org/wiki/Airplane#mediaviewer/File:Air_Berlin_B737-700_Dreamliner_D-ABBN.jpg

Thanks for Listening!

Questions?

References

- GeoMesa: https://www.geomesa.org
- “Spatiotemporal Indexing in Non-relational Distributed Databases”, Fox et al., http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6691586&isnumber=6690588
- Accumulo: https://accumulo.apache.org/
- LocationTech: https://www.locationtech.org
- BigTable: http://research.google.com/archive/bigtable.html