Interactive Visualization of Large-scale Movement Data using Apache Spark

Junjun Yin

CyberGIS Center for Advanced Digital and Spatial Studies
Department of Geography and Geographic Information Science
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign

April 5, 2016

Outline

- Big Movement Data
- Distributed computing environment in ROGER
  - Apache Spark
- Big Data processing in Hadoop Distributed File System (HDFS)
- Interactive data visualization with D3.js
  - Spark for data processing (Python) and D3 for visualization (JavaScript)

Outline – detailed

- Introduction to Spark
  - What is Spark, why and when to use Spark
  - Comparisons between Spark and Hadoop
  - Spark basics
- Visualization of density map of movement data
  - Density map generation for New York taxi GPS datasets
  - Implementation in Spark (based on space-time query)
- Visualization of movement flows from Twitter data
  - Workflow design
  - Data preparation
  - Mapping movement flows in real geographical space (e.g. using polygons in shapefile)
  - Summarizing the origin and destination networks
  - Interactive visualization using D3.js

Big Movement Data

- Let us recap on the different types of big movement data we have encountered in the previous lectures
  - New York taxi records
  - Location Based Social Media data (in particular, Twitter data)
- Challenges
  - e.g., large volume, messy, noisy
  - Can traditional GIS (geographical information system) work?

Visualization and knowledge discovery

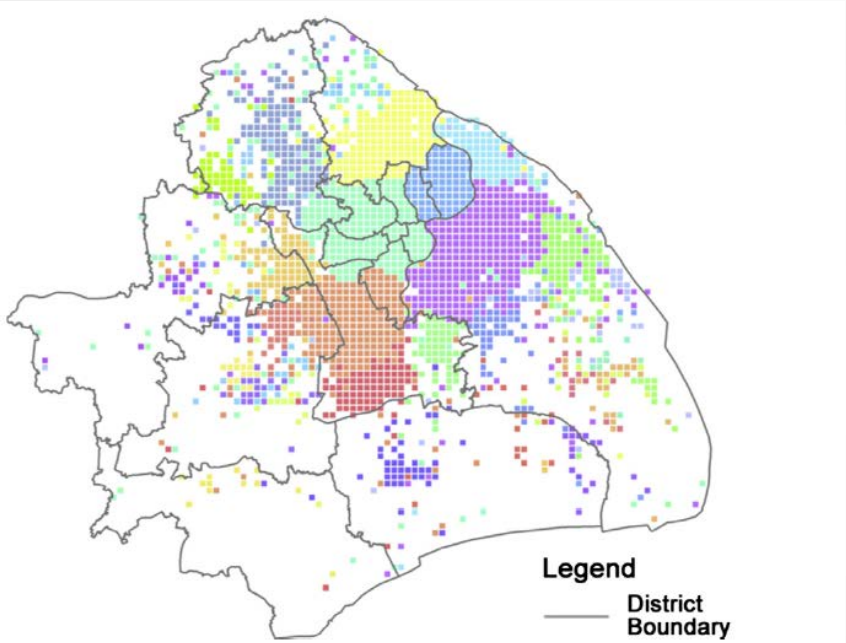
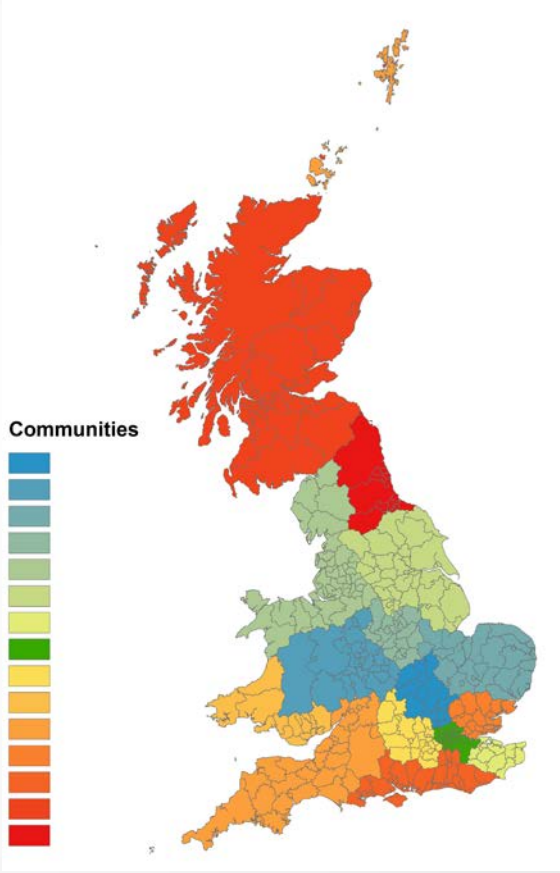


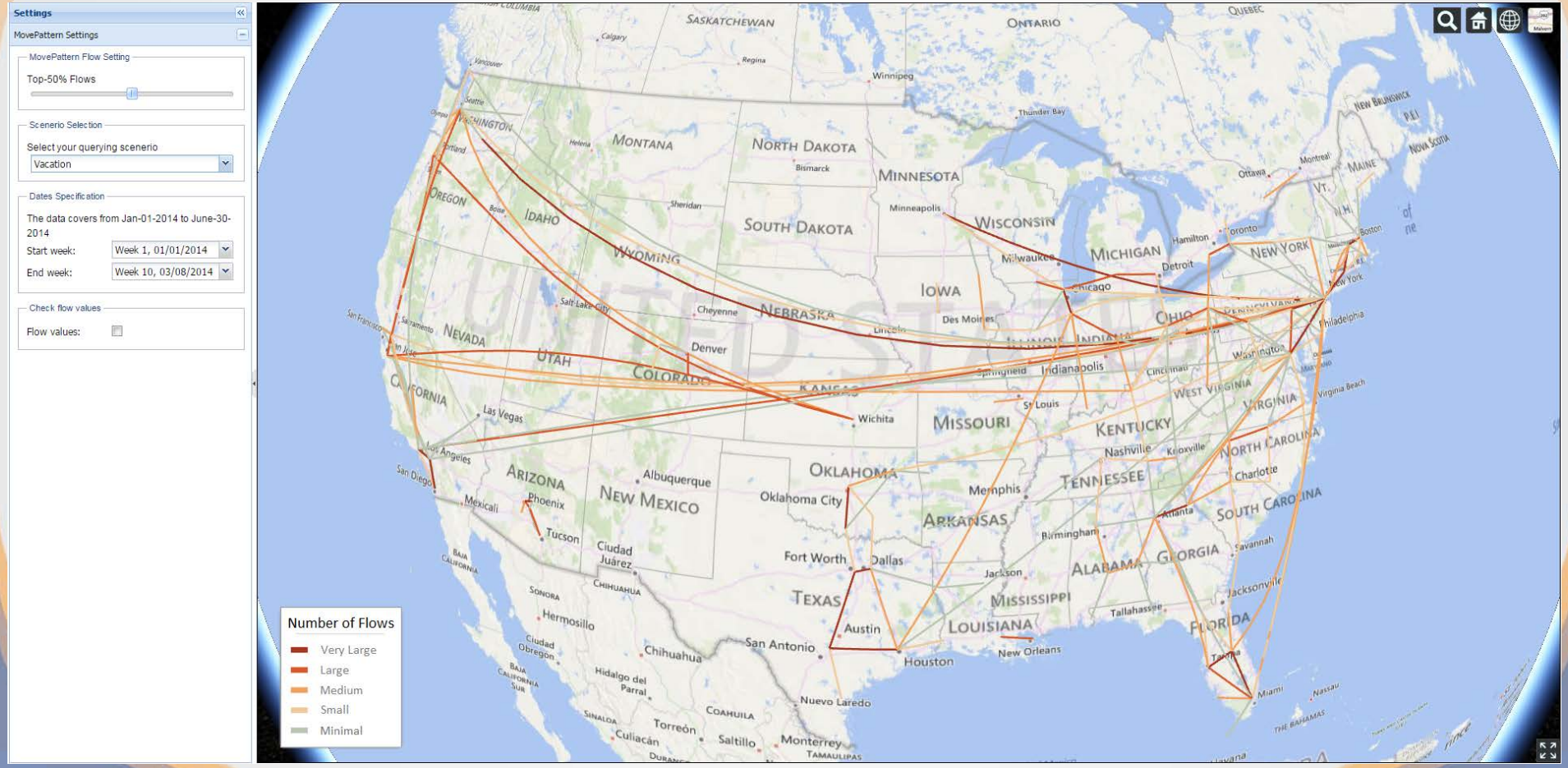
Fig. 3. Community detection result of the network constructed by all taxi trips. Cells in the same color are of the same community. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Source: X Liu, L Gong, Y Gong, Y Liu - Journal of Transport Geography, 2015



Community structure of the Great Britain based on the movement of Twitter users

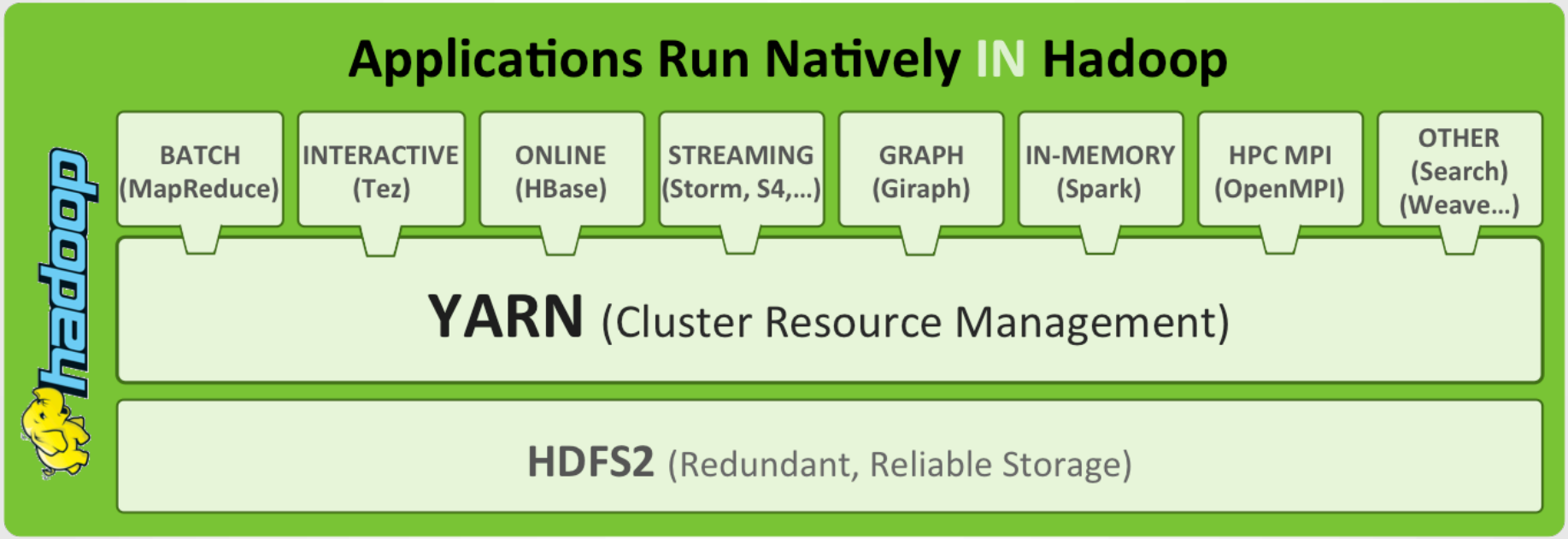
MovePattern



Distributed Computing with Spark

- What is Spark
  - http://spark.apache.org/
  - An open-source cluster computing framework originally developed in the AMPLab at UC Berkeley
  - The fundamental programming abstraction is Resilient Distributed Datasets (RDD), which is a logical collection of data partitioned across machines.
  - Run programs up to 100x faster Hadoop MapReduce in memory, or 10x faster on disk.
  - Ease of Use
    - Write applications quickly in Java, Scala, Python, R.
  - Runs Everywhere
    - Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.
- When to use Spark
  - When the operations involves iterations, especially machine learning and data mining algorithms
  - Repeatable/multiple queries on the same large dataset

Taking advantage of the Hadoop Architecture



Source: http://radar.oreilly.com/2014/01/an-introduction-to-hadoop-2-0-understanding-the-new-data-operating-system.html

Example in Spark

- launch spark:
  - >> ssh cg-hm08
  - >> module load java
  - >> pyspark

```
text_file = sc.textFile("words.txt")

counts = text_file.flatMap(lambda line:
line.split(",")).map(lambda word: (word,
1)).reduceByKey(lambda a, b: a + b)

#counts.saveAsTextFile("hdfs://...") (this saves to
HDFS format, we will not use it this time)

results = counts.collect()
print results
```

- type exit() to exit Spark shell

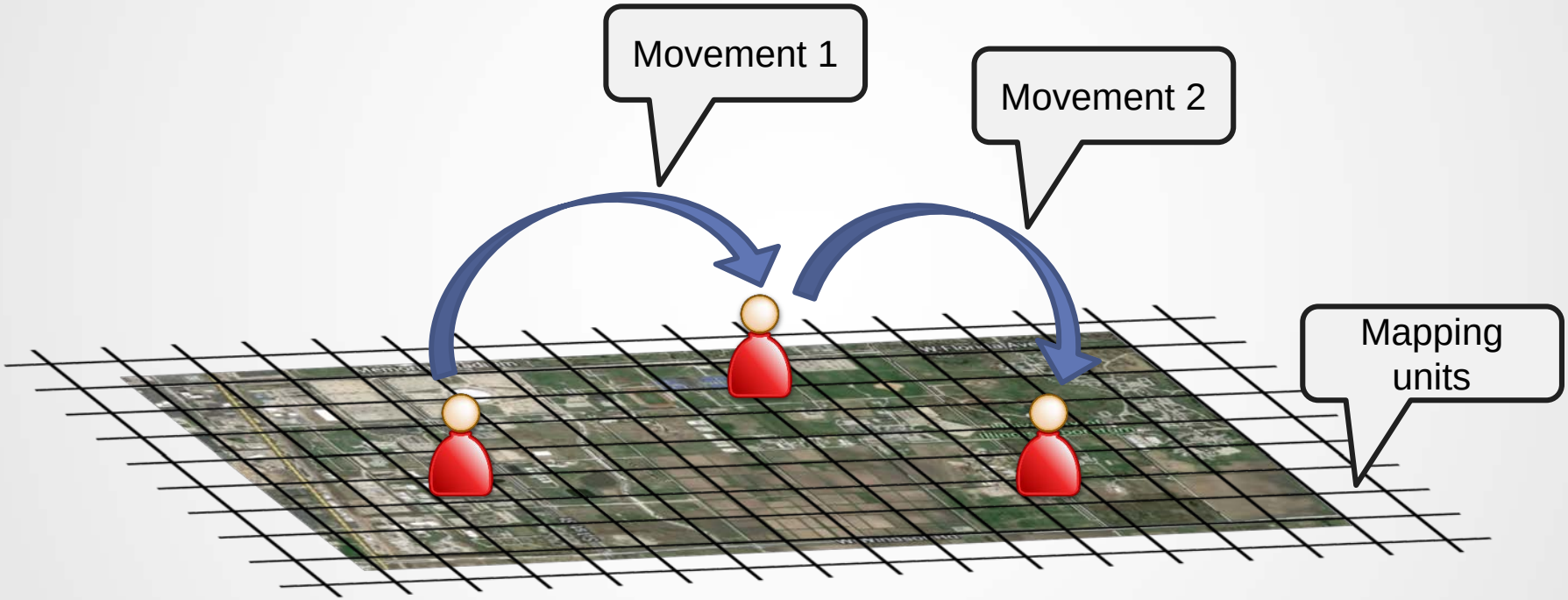
Basic Spark syntax

- pyspark script.py
- spark-submit script.py
- spark-submit --master yarn-cluster --executor-memory 20G --num-executors 50 script.py
- Important notes about spark-submit with YARN
- Set the parameters within the code

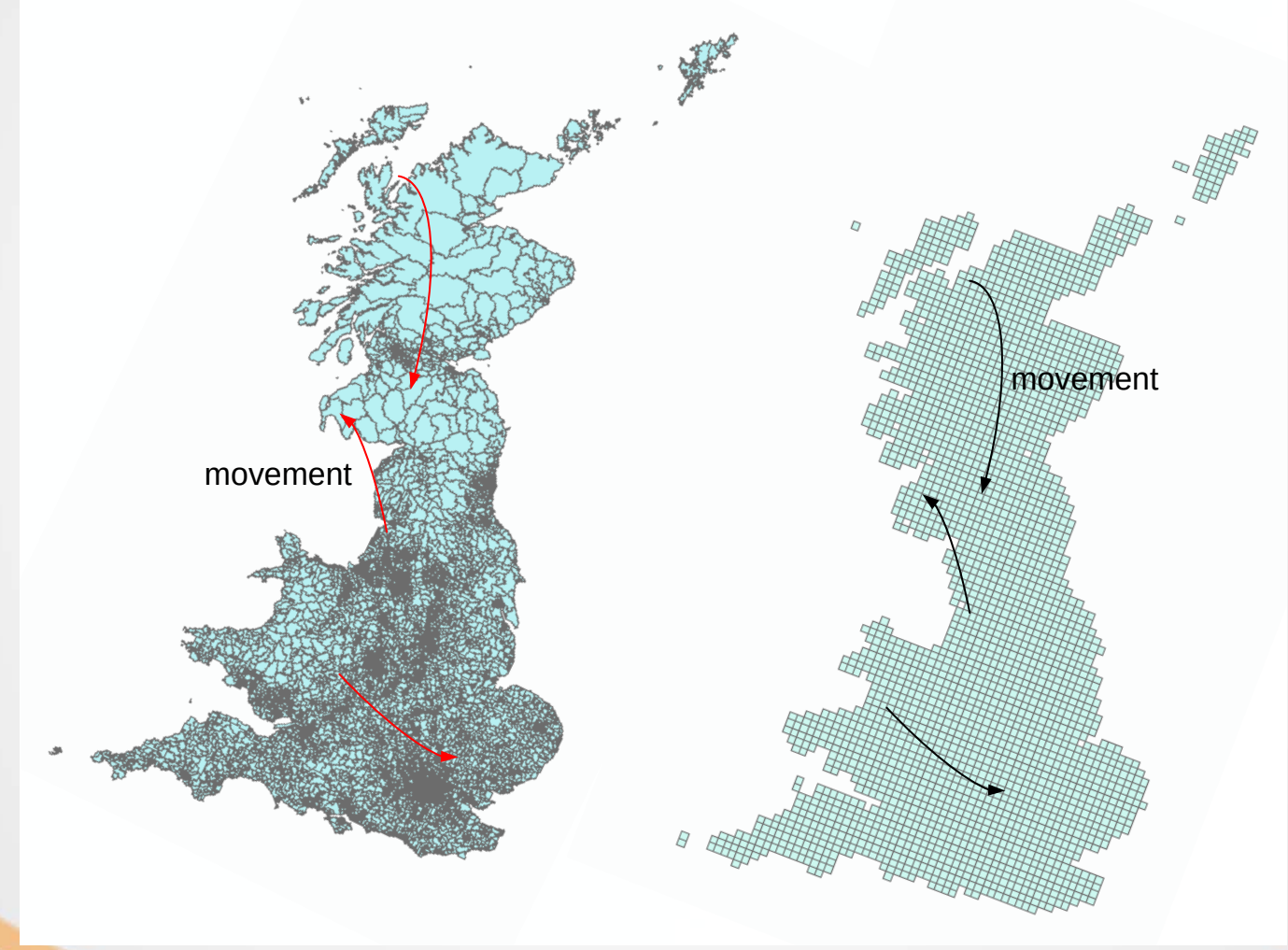
```
import pyspark
from pyspark.context import SparkContext
from pyspark import SparkConf, SparkContext
from pyspark.storagelevel import StorageLevel
 conf = (SparkConf().setMaster("yarn-client").setAppName("flow_generator").set("spark.executor.memory", "4g").set("spark.executor.instances", 50))
 sc = SparkContext(conf = conf)
```

Run the script using **spark-submit script.py**

Visualizing Twitter user movements



Illustrations of mapping movements in different units



(a) Polygon based mapping units (ward level) (b) Grid based mapping units (10 km)

Determine point-in-polygon Spark with ESRI's shapefile

- To map Twitter user's movement to a corresponding mapping unit, e.g. county, zip code area, state, etc., we need to perform "point-in-polygon" operation
  - Traditional spatial database provides such operation
  - For Spark, there is no spatial operation implemented nor does it recognize ESRI's shapefile and GeoJSON format
- The power of open source
  - For Python, there is Shapefile.py (https://github.com/GeospatialPython/pyshp)
  - Even Spark is open sourced
- Let's review how we did it with Hadoop and MapReduce

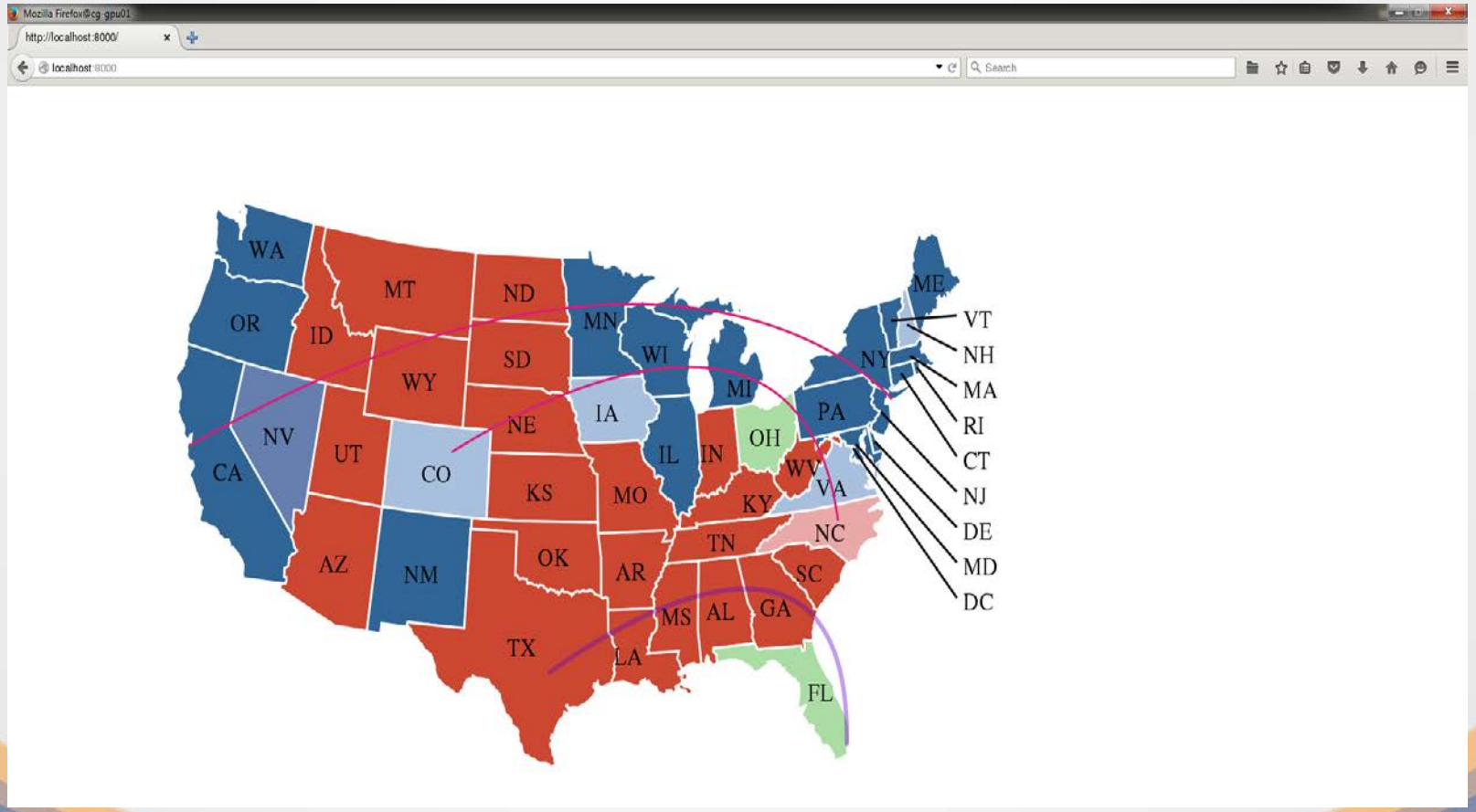
Determine point-in-polygon

- Review the script
  - How do we add files when submitting the job via YARN
  - How do we specify the parameters
  - How do we performs the searches to find the candidate polygon
  - How do we specify the input and output
  - How do we implement the filter() and map() functions
  - …
- What's next?
  - Now we know each point belong to a certain polygon, and we know the timestamp for which is the origin and which is the destination
  - The next is **mapping movement flows**

D3.js

- A JavaScript library for manipulating documents based on data.
- D3 provides visualization of data using HTML, SVG, and CSS.
- D3 combines powerful visualization components and a data-driven approach to DOM manipulation.
- Everything you want to know about D3.js, visit http://d3js.org/
- Check out the demos on the website
- For geographical visualization: http://datamaps.github.io/

Demo: Map of USA



Create your own map

- TopoJSON
- https://github.com/mbostock/topojson
- Usage:
- *topojson [options] -- [file ...]*
- Input files can be one or more of:
- .json GeoJSON or TopoJSON
- .shp ESRI shapefile
- .csv comma-separated values (CSV)
- .tsv tab-separated values (TSV)

Generate Twitter user movement flows

- For this workshop, let's consider the state-level Twitter user movements
- Suppose a user Twitter user tweeted at state A at time t1, and next moment, this user tweeted at state A at time t2, and next moment, at state B at time t3
- Therefore, the flow from A to B is one
- And imaging the collection of millions of Twitter users

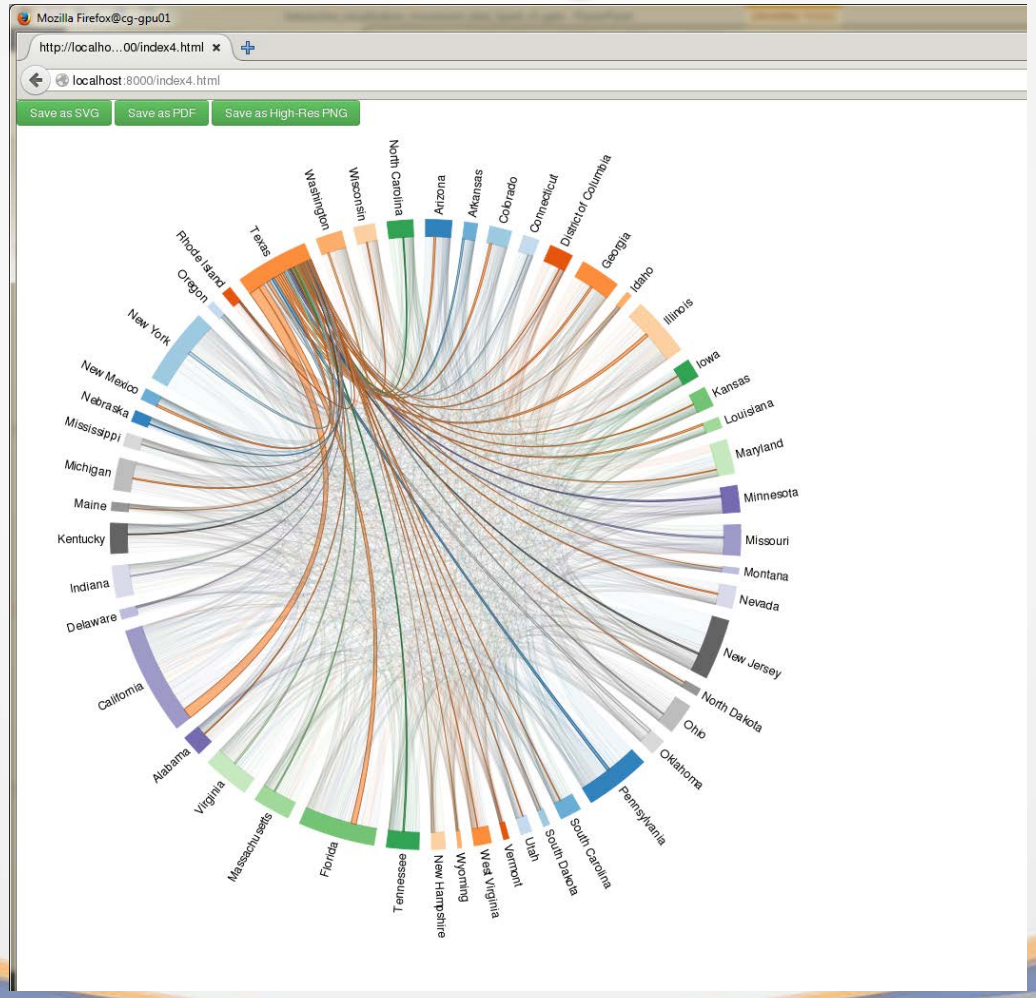
A, B, count1

B, C, count2

...

M, N, $count_n$

Chord diagram of Twitter user movement flows among states



Useful links

- General usage
  - (Spark Tutorial) http://lintool.github.io/SparkTutorial/
- Spatial Spark
  - (SpatialSpark) https://github.com/syoummer/SpatialSpark
  - (GeoSpark) http://geospark.datasyslab.org/