Taming Big Geospatial Data with Hadoop

Dr. 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, IL, 61801
jyn@illinois.edu

February 16, 2016

Outline

- Big Data
  - Geospatial Big Data
- CyberGIS
- Distributed computing environment in ROGER
  - Apache Hadoop
  - Apache Spark
- Big Data processing in Hadoop Distributed File System (HDFS)
  - MapReduce in Hadoop (Python) and Spark in Hadoop (Python)

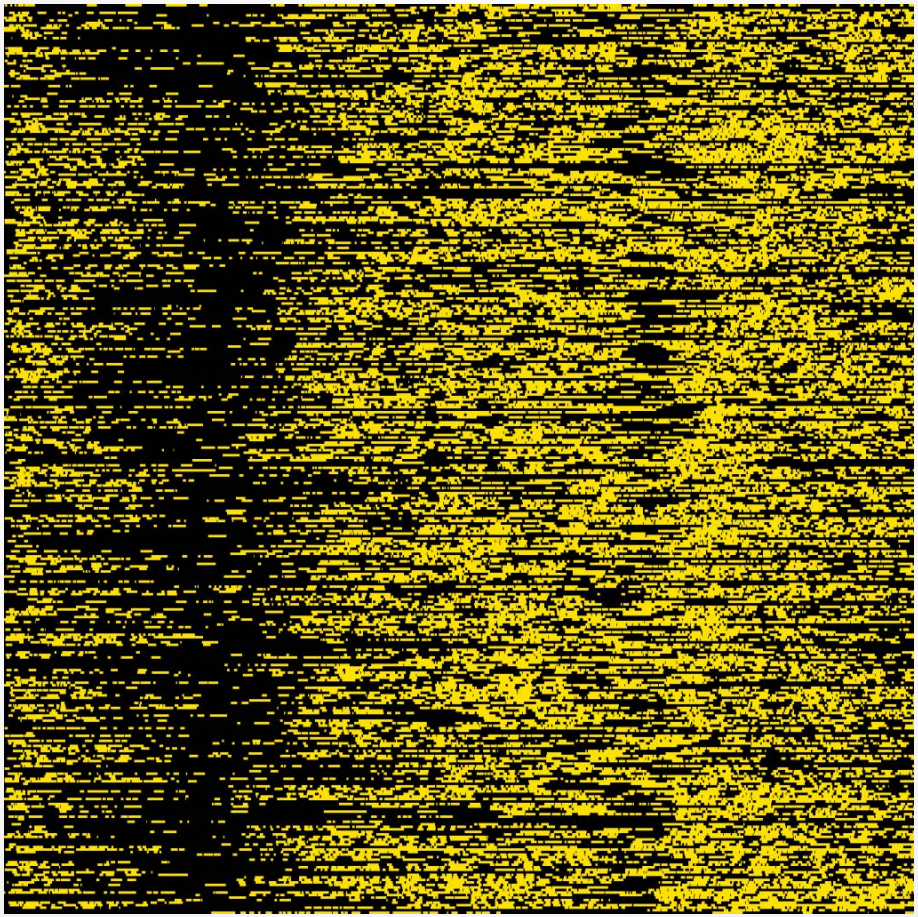
Outline – details

- Big Movement Data
  - Location Based Social Media Data (e.g. geo-located Tweets)
  - New York taxi data (demo)
  - Knowledge discover from movement datasets (e.g. movement flows and hot spots)
- CyberGIS
  - Getting to know CyberGIS
  - ROGER
  - Distributed computing environment for Big Data processing
- Introduction to Hadoop
  - MapReduce computing paradigm
  - Mapper, Reducer and combiner
  - HDFS (Hadoop File System)

Big Movement Data

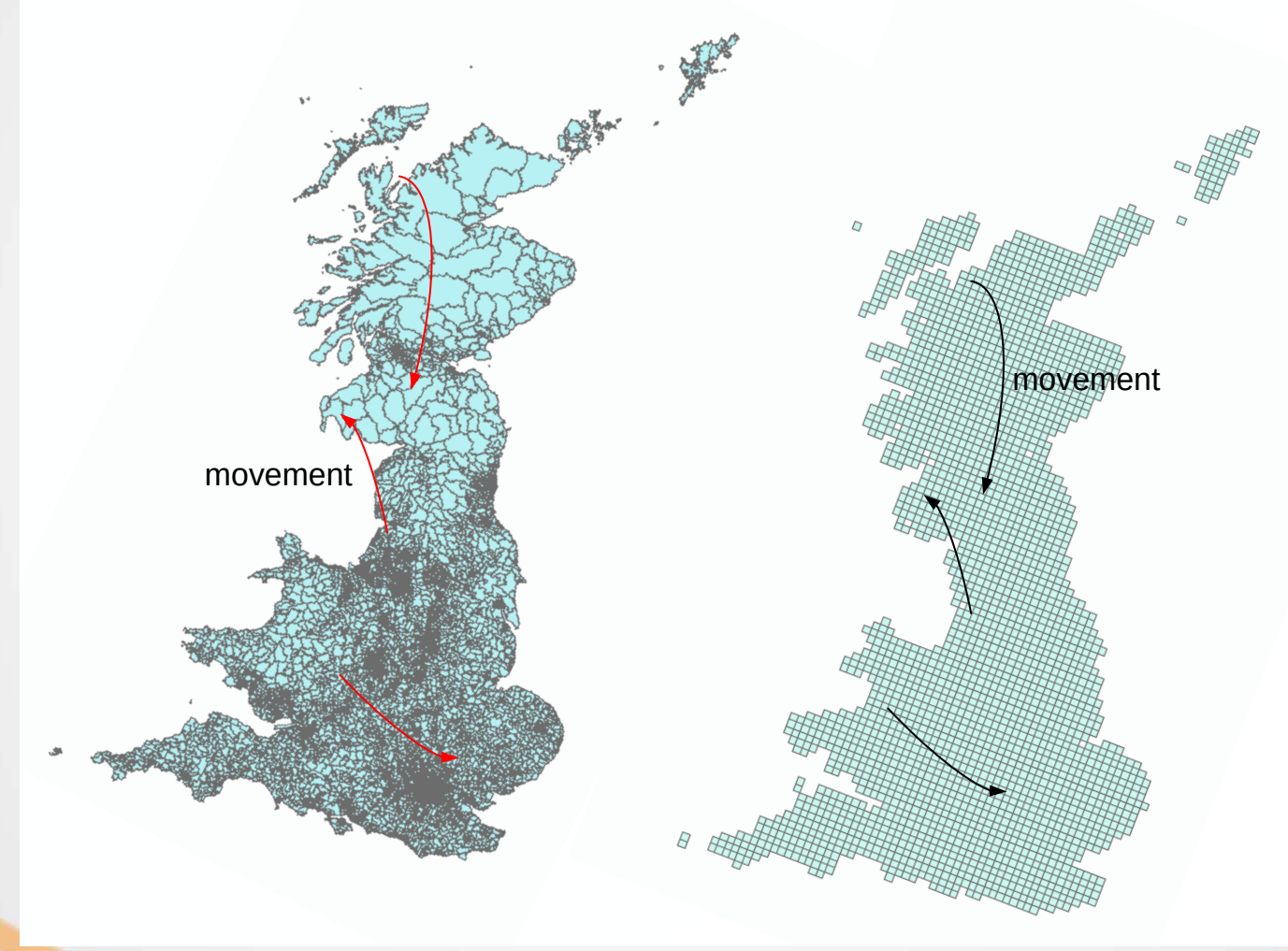
- In today's Big Data era, the advancement of positioning technology (e.g. GPS unit) enriches the data with spatial dimensions
- Internet of Things
  - Various types of sensors recording all activities with geo-tags and timestamps
  - Continuously
- Why spatial is special
  - Geo-located Twitter data (for Twitter user movements)
  - New York taxi data (for taxi movements)
- Challenges
  - e.g., large volume, messy, noisy
  - Can traditional GIS (geographical information system) work? (maybe?)
  - The need for cyberGIS for efficient data management and processing

Visualization and knowledge discovery



Visualization of New York taxi data
The black pixel means the taxi did not have fare and yellows means it did
Source: http://chriswhong.com/open-data/my-first-data-art-nyc-taxi-defrag/

Illustrations of mapping movements



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

Community detections from mobility patterns

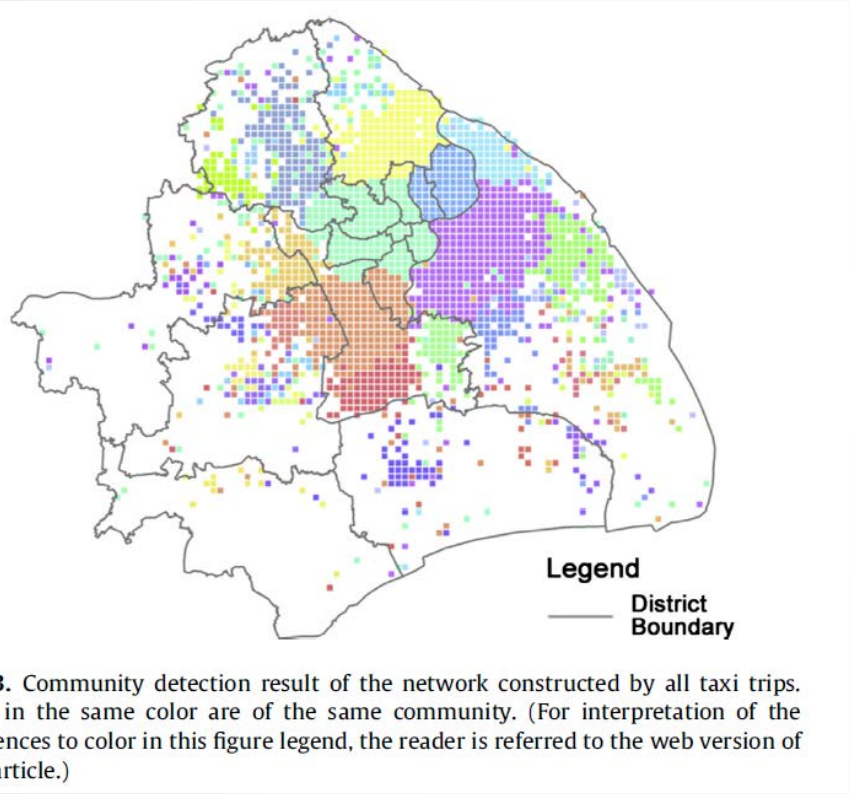
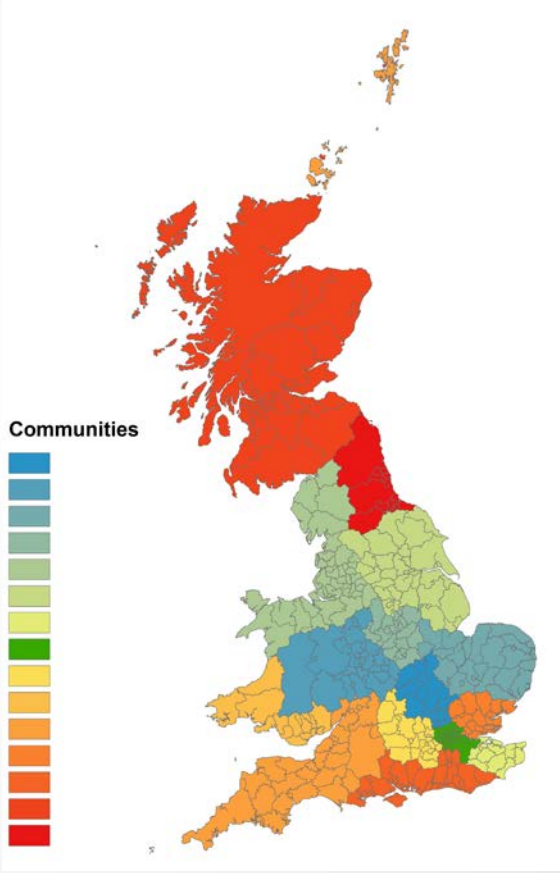


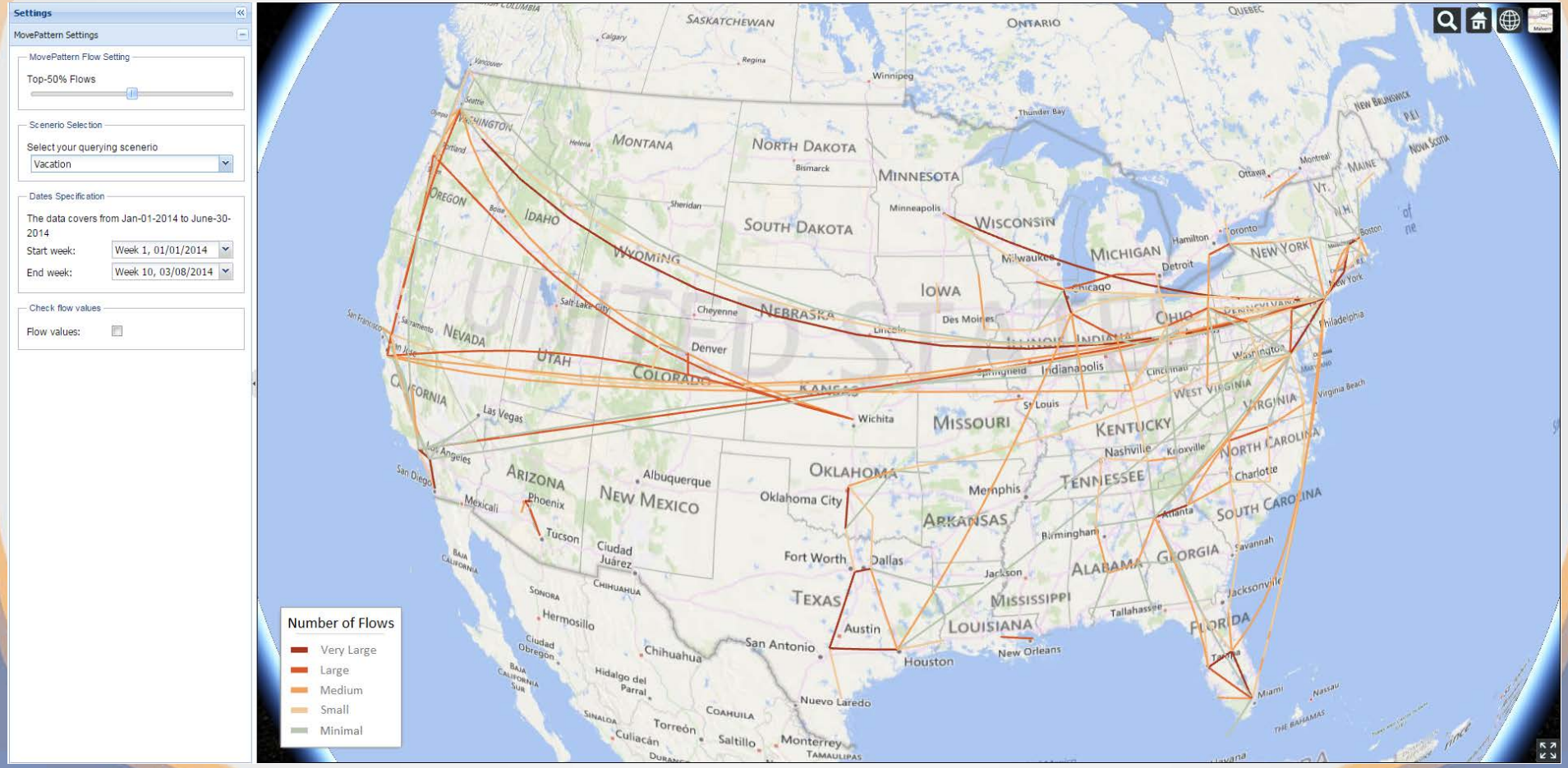
Fig. 3. Community detection result of the network constructed by all taxi trips. Cells in the same color are of the same community. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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 using Hadoop

- What is Hadoop
  - Distributed file system + replicating data in multiple nodes.
  - Easy-to-use MapReduce interface.
  - Scalable.
  - Parallel execution of mappers/reducers.
  - Fault tolerant.
- When to use Hadoop
  - Data is too big to fit into memory.
  - Tasks can be decomposed as batch-based processing.
  - The problem can be modeled by MapReduce computing paradigm.
  - Scalability is the major issue not interactivity.

Origins of MapReduce and Hadoop

- MapReduce is developed in Google
  - Dean, J. and Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), pp.107-113.
- Yahoo developed Hadoop
  - Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, May. The hadoop distributed file system. In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on (pp. 1-10). IEEE.
- Now Hadoop is an open source distributed computing framework, known as Apache Hadoop
  - http://hadoop.apache.org/

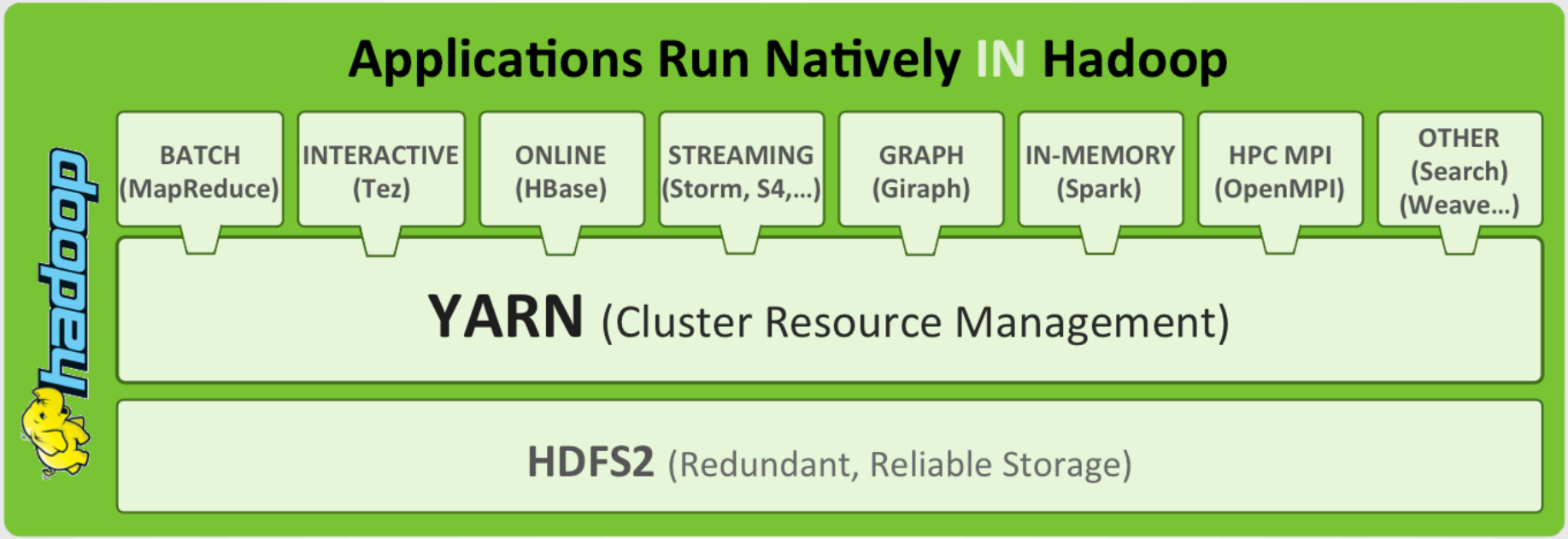
Characteristics of Hadoop

- Distributed file system + replicating data in multiple nodes.
- Easy-to-use MapReduce interfaces and libraries (e.g., Pig, Java, Hadoop Streaming API)
- Scalable.
- Parallel execution of mappers/reducers.
- Fault tolerant.
- Best practice for Hadoop
  - The problem can be modeled by MapReduce computing paradigm.
  - Scalability is the major issue not interactivity.

Hadoop in ROGER

- Dedicated Hadoop Cluster
  - Node cg-hm08 – cg-hm18
  - http://cg-hm08.ncsa.illinois.edu:50070/
- How to access
  - Login to ROGER first
    - ssh your_netID@roger-login.ncsa.illinois.edu
  - Login to the master node: cg-hm08
    - ssh cg-hm08

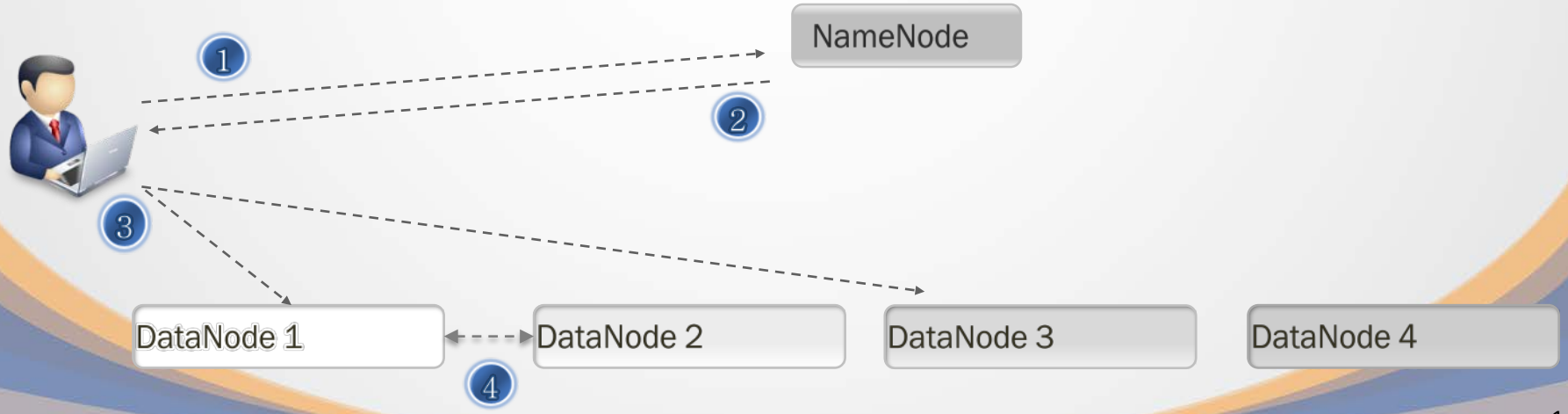
Hadoop Architecture



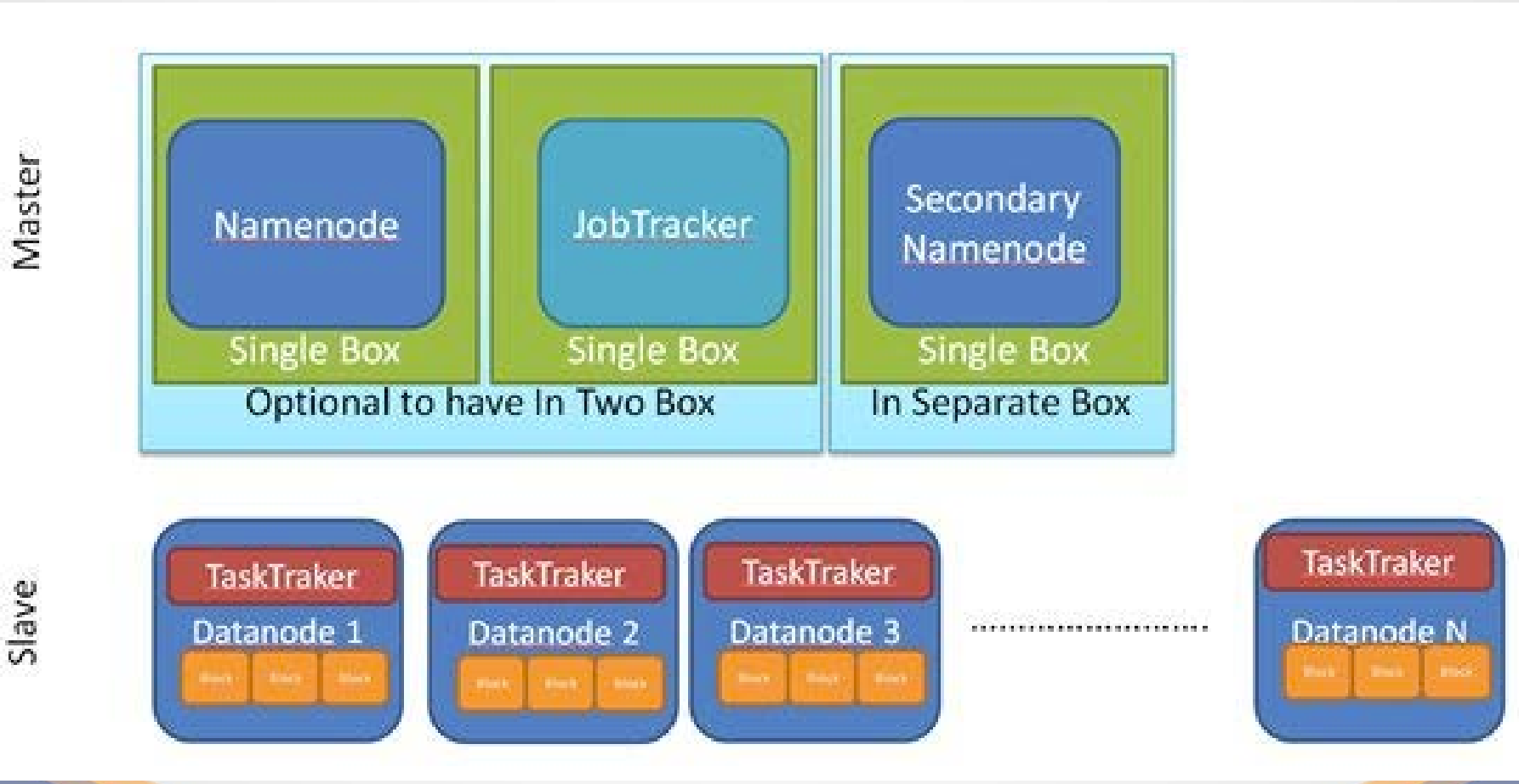
Source: http://radar.oreilly.com/2014/01/an-introduction-to-hadoop-2-0-understanding-the-new-data-operating-system.html

HDFS

- HDFS – Hadoop Distributed File System?
  - HDFS is the storage mechanism for Hadoop
  - Designed to scale easily and effectively
  - Designed to be deployed on low-cost hardware
- HDFS components
  - NameNodes (the master) manage the filesystem namespace
  - Datanodes (workers) store and retrieve the chunks of data



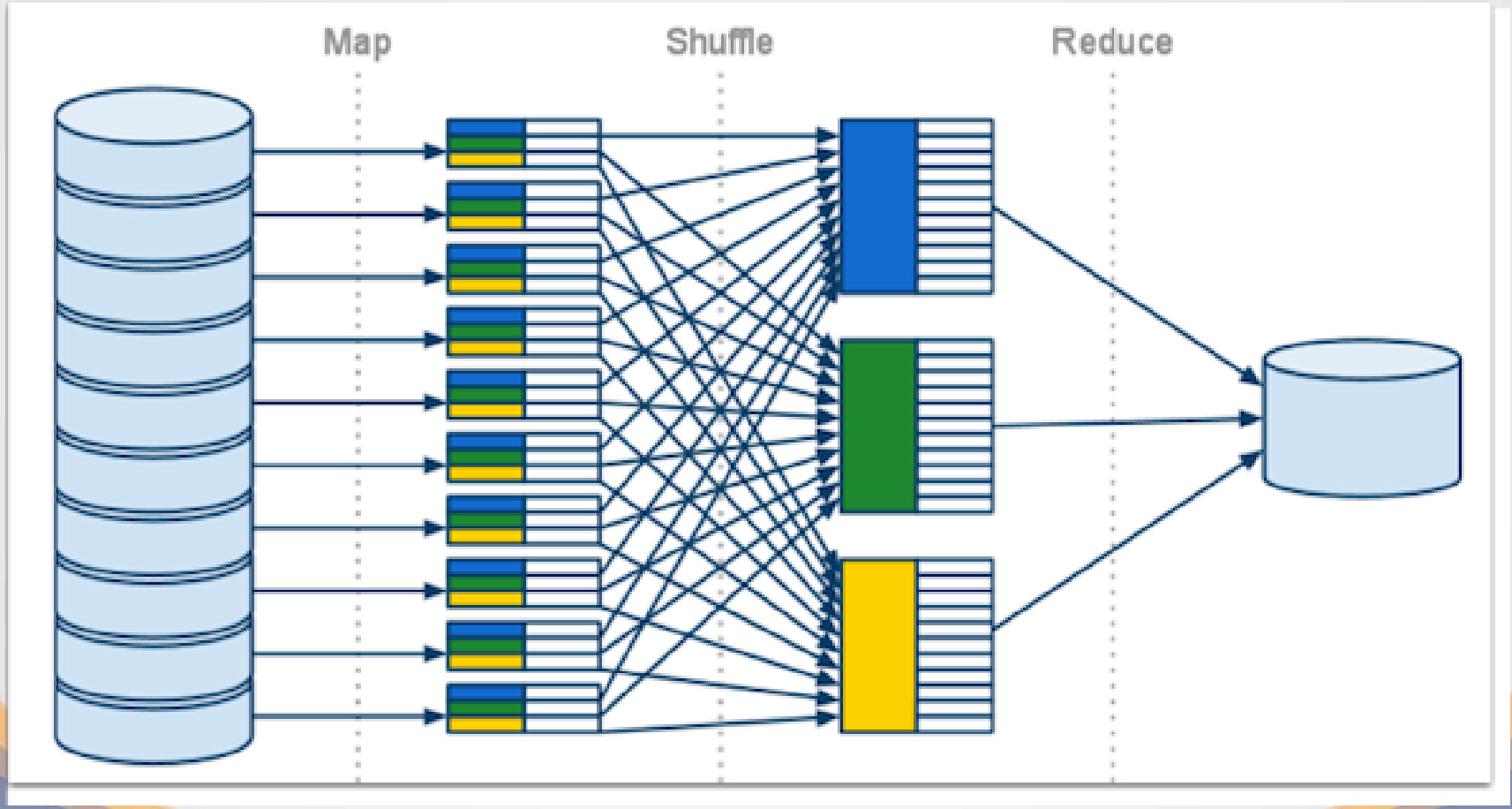
Terminologies



Terminologies

- **NameNode**
  - manages the file system metadata for HDFS
- **DataNode**
  - store the actual data
- **JobTracker**
  - responsible for scheduling the jobs' component tasks on the TaskTrackers, monitoring them and re-executing the failed tasks
- **TaskTracker**
  - execute the tasks as directed by the JobTracker
- **NameNode** and **JobTracker** are the masters

MapReduce computing paradigm



Example: word count in a corpus

- Find the frequency of words in a text corpus.
- Input: Bag of words =>
  - “Research”, “Student”, “GIS”, “CIGI”, “Student”, “Geospatial”, “Research”, “Center”, “GIS”, “Research”, “CIGI”, “Research”, “Geospatial”, “GIS”
- Output – <Word, Frequency>

Research	4
Student	2
CIGI	2

Center	1
GIS	3
Geospatial	2

Mapper

Mapper 1

Research
Student
GIS
CIGI
Student
Geospatial
Research

Key	Value
Research	1
Student	1
GIS	1
CIGI	1
Student	1
Geospatial	1
Research	1

Mapper 2

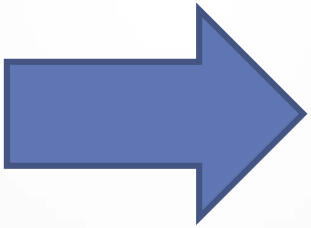
Center
GIS
Research
CIGI
Research
Geospatial
GIS

Key	Value
Center	1
GIS	1
Research	1
CIGI	1
Research	1
Geospatial	1
GIS	1

Shuffle

Key	Value
Research	1
Student	1
GIS	1
CIGI	1
Student	1
Geospatial	1
Research	1

Key	Value
Center	1
GIS	1
Research	1
CIGI	1
Research	1
Geospatial	1
GIS	1



CIGI	1
CIGI	1

Geospatial	1
Geospatial	1

GIS	1
GIS	1
GIS	1

Research	1
Research	1
Research	1
Research	1

Student	1
Student	1

Center	1

Reducer

CIGI	1
CIGI	1

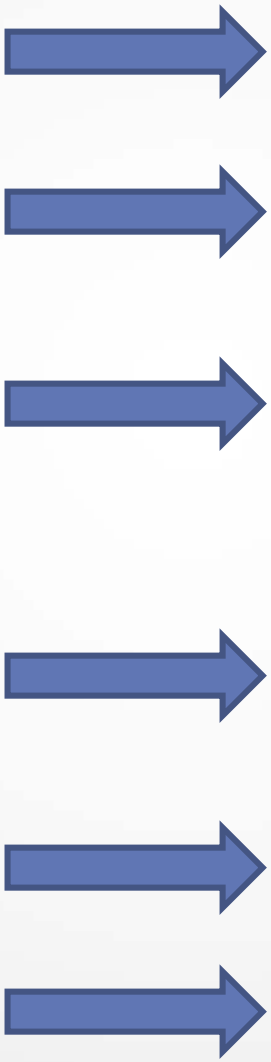
Geospatial	1
Geospatial	1

GIS	1
GIS	1
GIS	1

Research	1
Research	1
Research	1
Research	1

Student	1
Student	1

Center	1



Key	Value
CIGI	2

Key	Value
Geospatial	2

Key	Value
GIS	3

Key	Value
Research	4

Key	Value
Student	2

Key	Value
Center	1

Combiner

Mapper Output

Key	Value
Research	1
Student	1
GIS	1
CIGI	1
Student	1
Geospatial	1
Research	1

Key	Value
Research	2
Student	2
GIS	1
CIGI	1
Geospatial	1

Key	Value
Center	1
GIS	1
Research	1
CIGI	1
Research	1
Geospatial	1
GIS	1

Key	Value
Center	1
GIS	2
Research	2
CIGI	1
Geospatial	1

Reducer + Combiner

CIGI	1
CIGI	1

Key	Value
CIGI	2

Geospatial	1
Geospatial	1

Key	Value
Geospatial	2

GIS	1
GIS	2

Key	Value
GIS	3

Research	2
Research	2

Key	Value
Research	4

Student	2

Key	Value
Student	2

Center	1

Key	Value
Center	1

Basic Hadoop Commands

- Check Hadoop directory
  - hdfs dfs –ls [/user/netID/folder]
  - hdfs dfs –ls folder
- Make a Hadoop directory
  - hdfs dfs –mkdir folder
- Copy data into HDFS
  - hdfs dfs –copyFromLocal local_file [path_in_HDFS]
- Copy data from HDFS to local directory
  - hdfs dfs –getmerge file_in_HDFS path_in_local_directory
- Delete files from HDFS
  - hdfs dfs –rm –r file_in_HDFS

Example using Hadoop Streaming API

- Summarize the frequency of words occurred in the constitution of United States of America
- Code and data are stored in /gpfs_scratch/geog479/lecture5
- Copy the folder to your home directory
  - cp –r /gpfs_scratch/geog479/lecture5 ~/
  - cd lecture5/word_count_hadoop_python
  - hdfs dfs –copyFromLocal const.txt
  - hadoop jar /usr/hdp/2.3.2.0-2602/hadoop-mapreduce/hadoop-streaming-2.7.1.2.3.2.0-2602.jar -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input const.txt -output word_count.txt

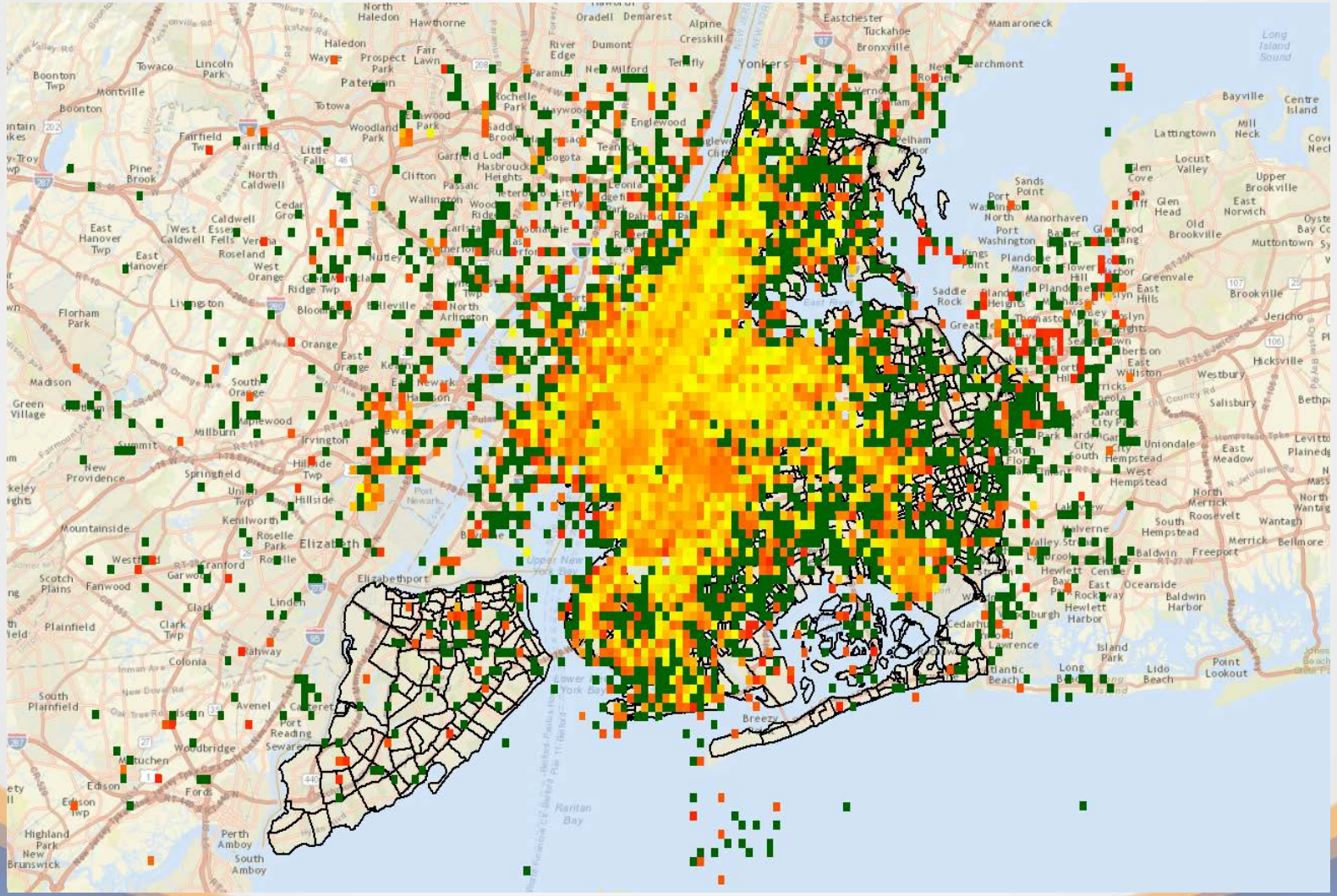
Example in Spark

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

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

counts = text_file.flatMap(lambda line:
line.replace(",", "").replace(".", "").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

New York taxi pickup passenger density map



Taxi Data

January 2013 Taxi Trips from FOIA (Freedom of Information Act) request from Chris Whong

(http://www.andresmh.com/nyctaxitrips/)

Trip:

```
medallion, hack_license, vendor_id, rate_code,
store_and_fwd_flag, pickup_datetime,
dropoff_datetime, passenger_count, trip_time_in_secs,
trip_distance, pickup_longitude, pickup_latitude,
dropoff_longitude, dropoff_latitude
```

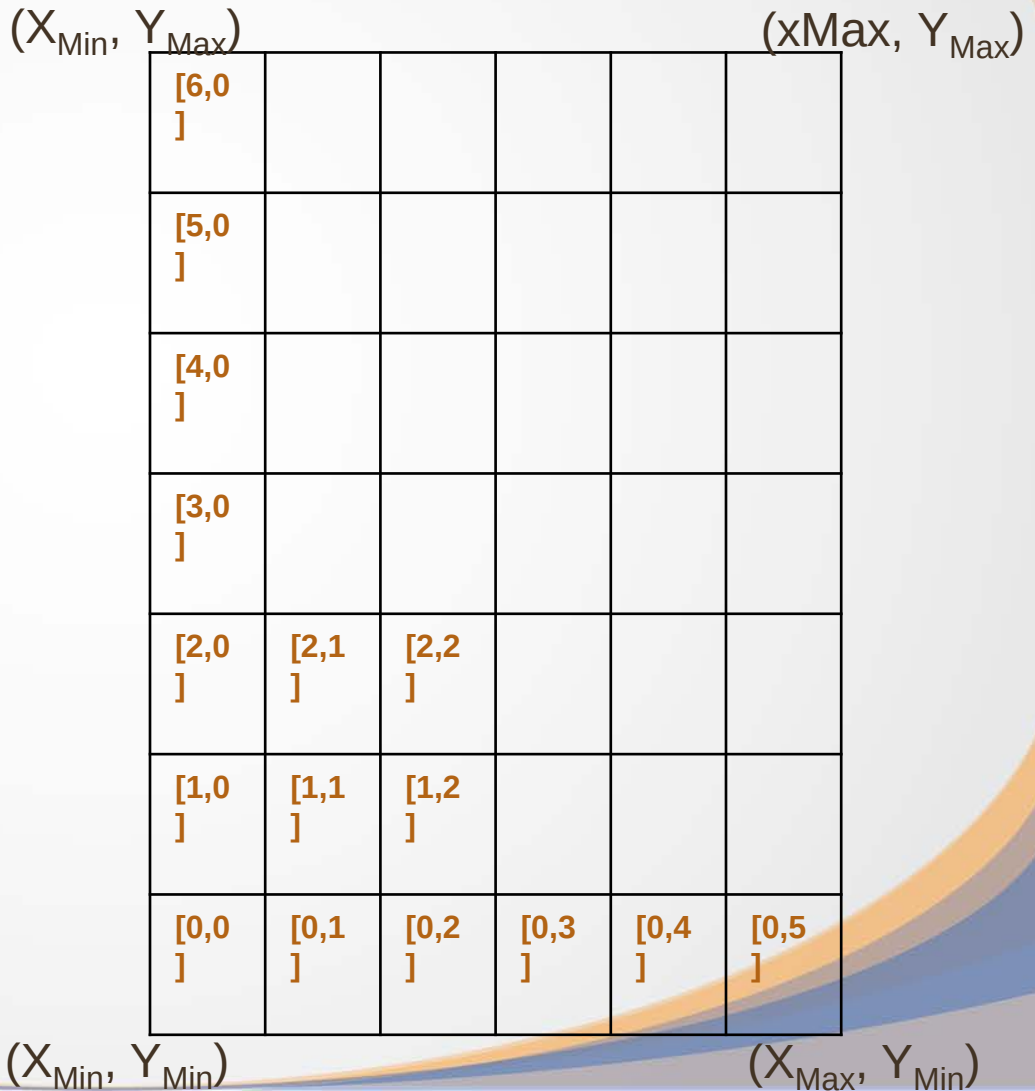
Fare:

```
medallion, hack_license, vendor_id, pickup_datetime,
payment_type, fare_amount,surcharge, mta_tax,
tip_amount, tolls_amount, total_amount
```

Find the cell number (index) that a point belongs to:

$$columnID = \text{floor}(\frac{X - X_{Min}}{cellSize})$$

$$\text{row}ID = \text{floor}(\frac{Y - Y_{Min}}{cellSize})$$



Quiz

- The client program that is uploading the data into HDFS performs I/O directly with the ( ).
- The ( ) only stores the metadata of the file system, but is not responsible for storing or transferring the data.
- A MapRedue job consists of three main phases: ( ), ( ), and ( )
- True or False: Files are write-once only (not updateable)

Apache Pig

This unit includes the following topics:

- What is Pig?
- Pig Script and Grunt Shell
- Pig Data Model
- Pig Latin and Relation Names
- Use Pig to navigate through HDFS and explore a dataset

What is Pig?

- Pig provides an engine for executing data flows in parallel on Hadoop[1].
- A scripting platform for processing and analyzing large data sets
  - Apache Pig allows Apache Hadoop users to write complex MapReduce transformations using a simple scripting language called Pig Latin.
  - Pig Latin includes operators for many of the traditional data operations (join, sort, filter, etc.), as well as the ability for users to develop their own functions for reading, processing, and writing data.
- How Pig differs from MapReduce?
  - It can do early error checking (did the user try to add a string field to an integer field?) and optimizations (can these two grouping operations be combined?).
  - Pig Latin cost less to write and maintain than Java code for MapReduce.
  - Multiple operators are provided.

1. Gates, A. (2011). *Programming Pig*. " O'Reilly Media, Inc.".

Apache Pig Philosophy

- The Apache Pig Project published founding philosophy for pig developers[1].
- Pigs Eat Anything
  - Pig can process any data, structured or unstructured.
- Pigs Live Anywhere
  - Pig is not just for Hadoop. Pig can run on any parallel data processing framework.
- Pigs Are Domestic Animals
  - Pig is designed to be easily controlled and modified by its users.
- Pigs Fly
  - Pig processes data quickly.

1. http://pig.apache.org/philosophy.html

Pig Script and Grunt Shell

- Pig Latin scripts can be executed by Pig script or Grunt shell
- Pig script
  - Write a Pig Latin program in a text file and execute it using the **pig** command

```
[cgtrn20@cg-hm06 Exe1]$ pig filter.pig
```

- Grunt shell
  - Grunt is Pig's interactive shell. It enables users to enter Pig Latin interactively and provides a shell for users to interact with HDFS.

```
[cgtrn20@cg-hm06 Exe1]$ pig
15/07/27 14:12:55 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
15/07/27 14:12:55 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
15/07/27 14:12:55 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2015-07-27 14:12:55,582 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14
.0.2.2.0.0-2041 (rexported) compiled Nov 19 2014, 15:24:46
2015-07-27 14:12:55,583 [main] INFO  org.apache.pig.Main - Logging error messages
to: /gpfs/smallblockFS/home/cgtrn20/cybergis/cybergis/Exe1/pig_1438024375580.log
2015-07-27 14:12:55,618 [main] INFO  org.apache.pig.impl.util.Utils - Default boot
up file /home/cgtrn20/.pigbootup not found
2015-07-27 14:12:56,545 [main] INFO  org.apache.pig.backend.hadoop.executionengine
.HExecutionEngine - Connecting to hadoop file system at: hdfs://cg-hm03.ncsa.illin
ois.edu:8020
grunt>
```

Pig's Data Model

- Pig's data types are consisted of two categories:
  - Scalar types contain a single value
  - Complex types contain other types.
- Scalar Types
  - Int, long, float, double, chararray (strings of Unicode characters), bytearray (a blob), boolean
- Complex types
  - Map: Collection of key value pairs
    - [name#David,department#CyberGis,zip#61821]
  - Tuple: Ordered set of fields
    - (David,CyberGIS,61821) – Tuples are divided into fields.
  - Bag: Unordered collection of tuples
    - {(David,CyberGIS,61821), (Tom,Mathmatics,61820), (Clair,Computer Science,61821) }

Pig Latin 1

- Pig Latin is a dataflow language.
- Each processing step results in a new data set, or relation.
  - input= LOAD ‘twitterdata.txt' USING PigStorage(',')
  - In input = LOAD ’ twitterdata.txt ', input is the name of the dataset in HDFS.
- Case sensitive
  - Pig Latin cannot decide whether it is case-sensitive.
  - Keywords in Pig Latin are not case-sensitive (e.g., LOAD is identical to load)
- Comments
  - SQL-style single-line comments (--) and Java-style multiline comments (/* */).
- Pig Latin Reference Manual
  - http://pig.apache.org/docs/r0.7.0/piglatin_ref2.html

Pig Latin 2

- LOAD
  - raw = LOAD 'cybergis/radiation.csv' USING PigStorage(',') AS (CapturedTime : chararray,Latitude:double,Longitude:double,Value:int,Unit:chararray);
  - Pig is very lenient in terms of schemas:
    - If you define a schema, then Pig will perform error-checking with it.
    - If you do not define a schema, Pig will make a best guess as to how the data should be treated
- STORE
  - STORE result into 'heatmap_output';
  - The store command sends the output to a folder in HDFS.
- DUMP
  - DUMP result;
  - The dump command directs the output of your script to your screen.

Pig Latin 3

- FILTER
  - **step1 = FILTER step0 BY $4 MATCHES 'cpm';**
  - Use FILTER to select the rows you want or filter out the rows you do not want.
  - Select the rows by its fifth field (i.e., $4).
- FOREACH…GENERATE
  - **step4 = FOREACH step3 GENERATE $CapturedTime,$3,$4;**
  - The FOREACH operator transforms your data into different data sets.
  - The FOREACH takes a set of expressions and applies them to every record in the data pipeline,
- GROUP
  - **step5_group = GROUP step5 BY (latkey,logkey);**
  - The group statement collects together records with the same key.
- LIMIT
  - **L = LIMIT step5_group 10;**
  - See only a limited number of results.

Preparing the data using Pig

- PIG is a higher level, SQL-like programming language for Hadoop jobs.
- Easier code for simple (and even some advanced) Hadoop jobs.
- On a compute cluster, it will be around the same speed as the Java MapReduce code
- In this demonstration, we will only use one day data (1st February, 2014)
  - ssh net_ID@roger-login.ncsa.lllinois.edu
  - ssh cg-hm08
  - cd ~/lecture5/pig
  - hdfs dfs –copyFromLocal day1.txt [user/file_name]
- The Pig script is located in:
  - cd ~/lecture5/pig
  - pig -f filter_usa.pig -param input=day1.txt

An example geo-located Twitter message

"**user_id**"Some little bug has bitten me in the night and my knee is the size of a melon at present.. Not the best of starts to my day! #antihystermine**51.3432617,1.4025932**Twitter for Androidfalse1**Tue Jul 01 23:40:46 CDT 2014**-1en166145388http://abs.twimg.com/images/themes/theme10/bg.gifLe5leyH25b176c519f167e6BroadstairsBroadstairsUnited Kingdomneighborhoodnull(51.326601,1.3832097),(51.377546,1.3832097),(51.377546,1.449228),(51.326601,1.449228),Polygonhttps://api.twitter.com/1.1/geo/id/25b176c519f167e6.jsonLesley HopperRamsgate. Kent. England2082203759Tue Jul 13 07:55:49 CDT 2010

User_ID, latitude1, longitude1, timestamp1

↓

User_ID, latitude2, longitude2, timestamp2

↓

Trajectory: User_ID, <location1, timestamp1>, <location 2, timestamp2>, …

Pig script

```
raw = LOAD '${input}' USING PigStorage('\u0001') AS
(tweetId,text:chararray,geo,source,isretweet,search_id,create_at,meta_
data,to_user_id,language_code,user_id,profile_image_url,user_name,plac
e_id,place_name,place_fullname,country,place_type,street_address,bound
ary,boundary_type,place_url);

clean = DISTINCT raw;

step0 = FILTER clean BY geo is not null AND geo != 'null' AND  geo !=
'' AND isretweet == 'false' AND create_at is not null AND create_at !=
'null' AND create_at != '';

step1 = FOREACH step0 GENERATE user_id, flatten(STRSPLIT(geo, ',')),
To_ISO(Get_Time(REPLACE(REPLACE(create_at, 'CDT', '-05:00'), 'CST', '-
06:00'), 'EEE MMM dd HH:mm:ss Z yyyy')) , text;

-- $2 is longitude, $1 is latitude
step2 = FILTER step1 BY $2 > -167.276413 AND $2 < -56.347517 AND $1 >
5.499550 AND $1 < 82.296478;

step3 = FOREACH step2 GENERATE $0, $1, $2, $3;
STORE step3 INTO '${input}_filtered.txt' USING PigStorage(',');
```

Preparing the data using Pig

- We will revisit the script and practice in this week's lab