Taming Big Geospatial Data with Hadoop re-visit

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 23, 2016

Apache Pig

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

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.".

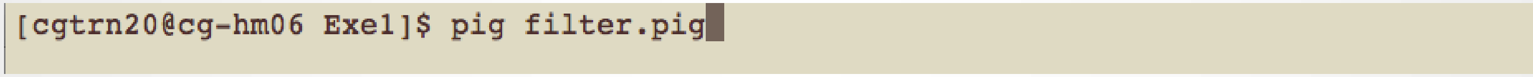
Apache Pig Philosophy

- 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

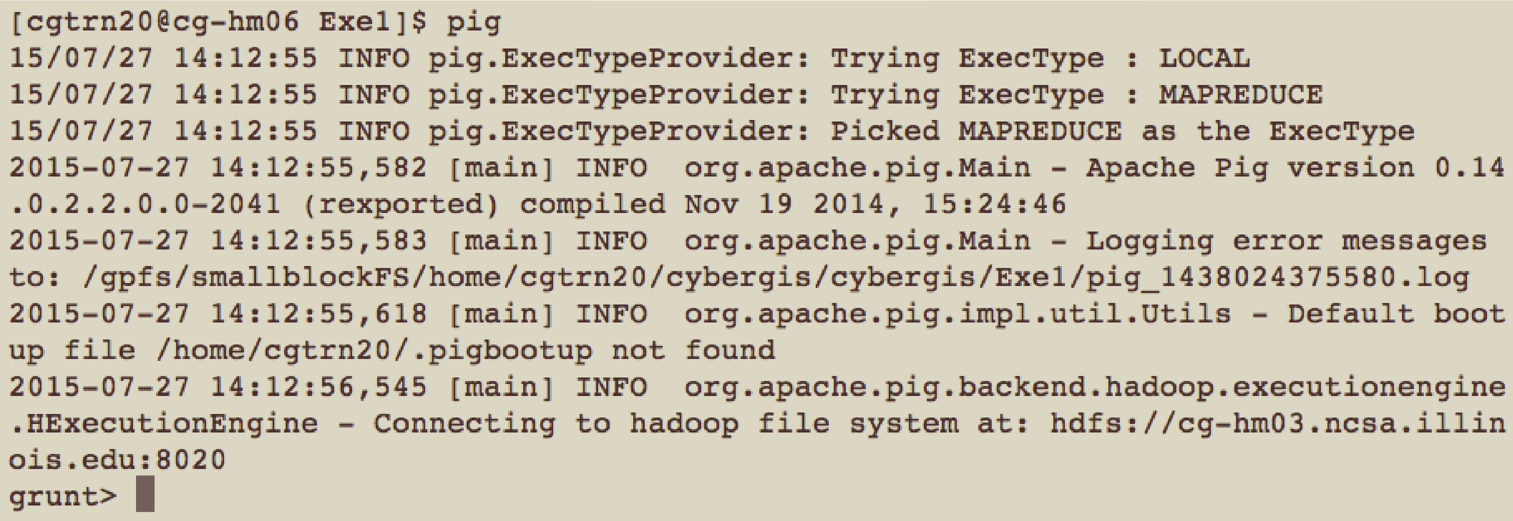
Pig Script and Grunt Shell

- 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

- ssh net_ID@roger-login.ncsa.Illinois.edu
- ssh cg-hm08
- cd ~/lecture6/pig
- hdfs dfs -copyFromLocal tweets.txt [user/file_name]

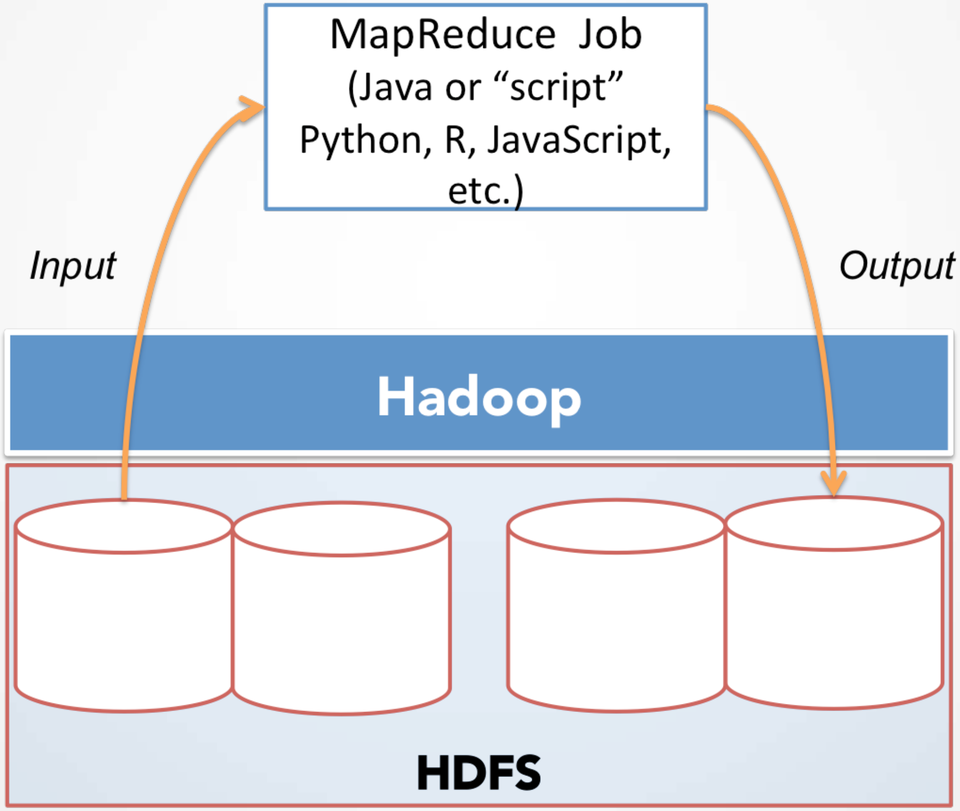
- The Pig script is located in:
  - cd ~/lecture6/pig
  - pig simple.pig

Pig script

```
raw = LOAD ‘input_file_name' 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, ',')),
REPLACE(REPLACE(create_at, 'CDT', '-05:00'), 'CST', '-06:00'), 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 ‘output_filename' USING PigStorage(',');
```

Hadoop Streaming API

Delegation of Map and Reduce tasks



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/lecture6
- Copy the folder to your home directory
  - cp -r /gpfs_scratch/geog479/lecture6 ~/
  - cd lecture6/word_count_hadoop_python
  - hdfs dfs –copyFromLocal const.txt

Mapper

```
#!/usr/bin/env python
import sys

### This is a great day (yes, a great day), but we are sitting inside doing coding.

for line in sys.stdin:
    line =
line.strip().replace('(','').replace(')','').replace(':','').replace('.','').replace(';','').replace(',','')

    words = line.split()
    #what does "word" look like

###remember, this is a mapper, the sole task is to prepare key and value pairs to reducer
    for word in words:
        print '%s,%s' % (word, 1)
```

Test the mapper

```
echo " This is a great day (yes, a great day), but we are sitting inside
doing coding" | lecture6/word_count_hadoop_python/mapper.py
```

```
This,1
is,1
a,1
great,1
day,1
yes,1
of,1
course,1
but,1
we,1
are,1
sitting,1
inside,1
doing,1
coding,1
```

Reducer

- For reducer, you can even choose other programing language other than Python
- By default, reducer assumes they are operating on the same key
  - However, in Hadoop Streaming API, we have to determine whether the input to reducer belong to the same key.
- Check out the reducer.py for details
- Check out the reducer2.py for another implementation

Test the reducer

```
echo " This is a great day (yes, a great day), but we are sitting inside doing coding " | ~/lecture6/word_count_hadoop_python/mapper.py | ~/lecture6/word_count_hadoop_python/reducer.py
```

```
a,2
great,2
sitting,1
doing,1
inside,1
coding,1
we,1
but,1
This,1
are,1
yes,1
day,2
is,1
```

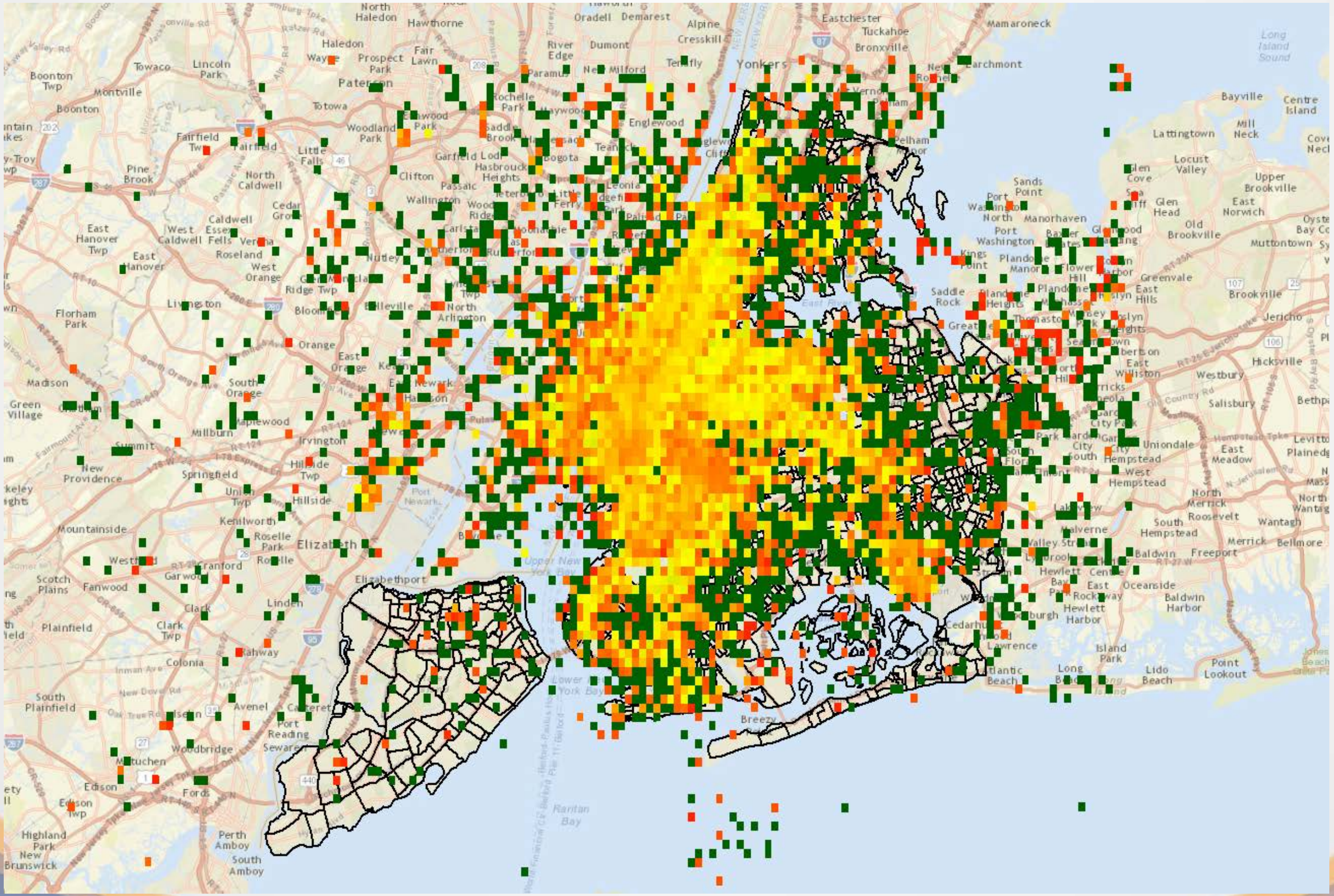
Run Hadoop Streaming Job

- **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
- or
- **hadoop jar** /usr/hdp/2.3.2.0-2602/hadoop-mapreduce/hadoop-streaming-2.7.1.2.3.2.0-2602.jar **-files** word_count_hadoop_python **-mapper** “word_count_hadoop_python/mapper.py” **-reducer** “word_count_hadoop_python/reducer.py” **-input** const.txt **-output** word_count.txt

Quiz

- How to check out the results?

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
```

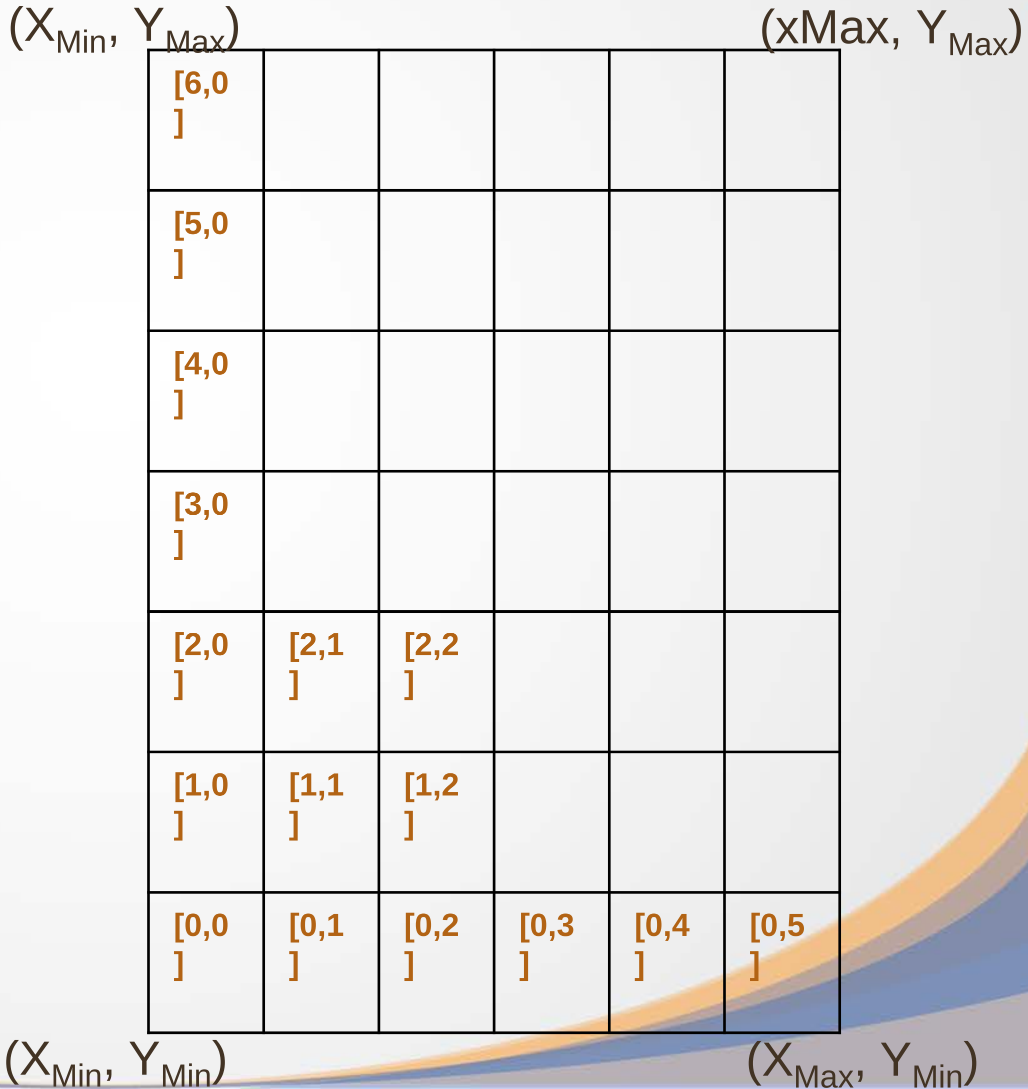
Fare:

```
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:

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

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



[6,0]					
[5,0]					
[4,0]					
[3,0]					
[2,0]	[2,1]	[2,2]			
[1,0]	[1,1]	[1,2]			
[0,0]	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]

Implementation

- First, let's be aware that we are designing a MapReduce program
- What should the Mapper do?
  - What is the key now?
  - And what is the value to the key?
- What should the Reducer do?
  - If they belong to the same key, what would you do