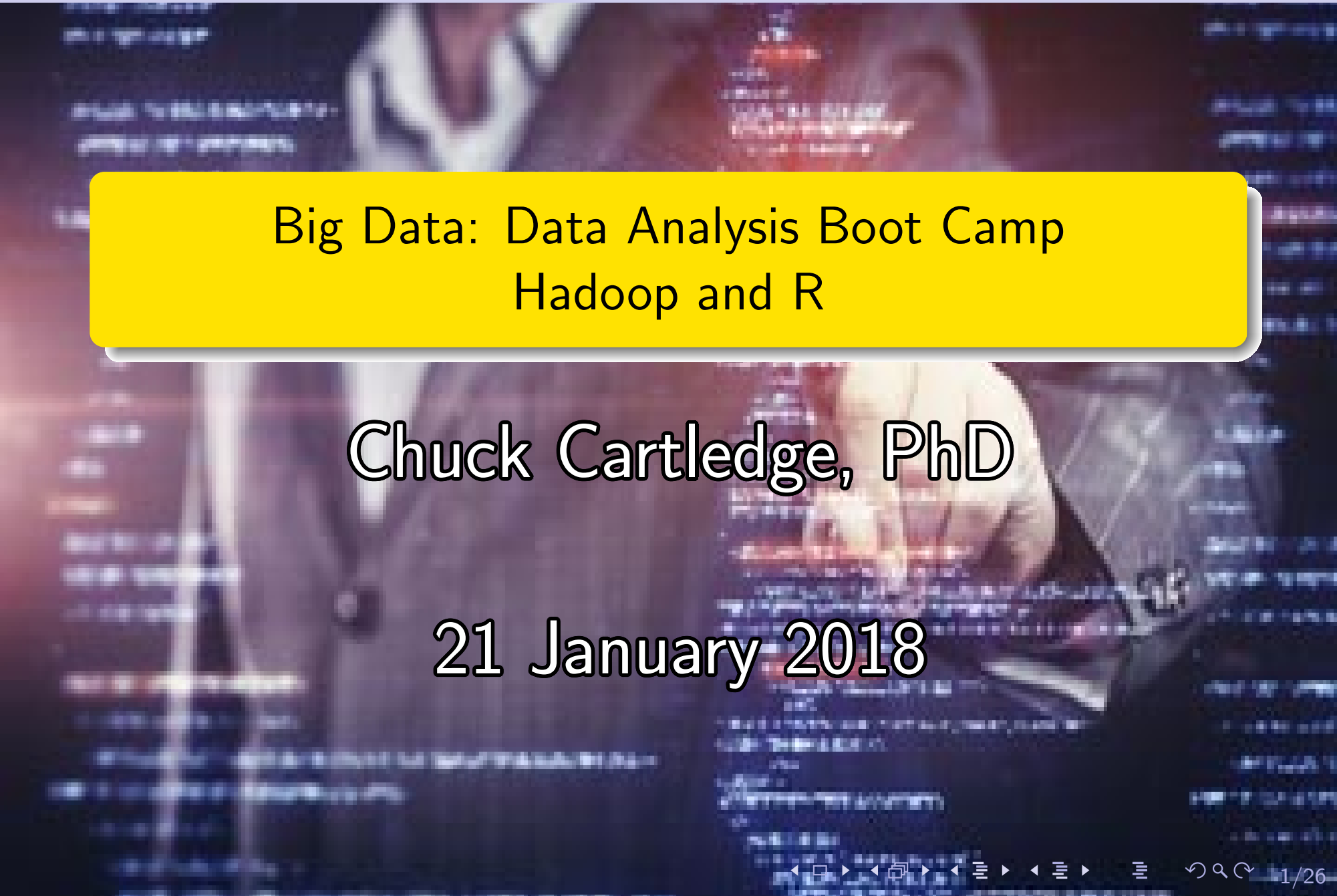


Big Data: Data Analysis Boot Camp
Hadoop and R

Chuck Cartledge, PhD

21 January 2018

Table of contents (1 of 1)

What are we going to cover?

1. Look at the Hadoop map-reduce programming model
2. Pick apart the "classic" map-reduce word count program
3. Look at how the map-reduce model can be used with complex keys

The Hadoop Distributed File System (HDFS)

"The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is now an Apache Hadoop subproject."

A. Staff [2]

HDFS Assumptions and Goals[2]

Hardware Failure Hardware failure is the norm rather than the exception.

Streaming Data Access Applications that run on HDFS need streaming access to their data sets.

Large Data Sets Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size.

Simple Coherency Model HDFS applications need a write-once-read-many access model for files.

Moving Computation is Cheaper than Moving Data A computation requested by an application is much more efficient if it is executed near its data.

Portability Across Heterogeneous Hardware and Software Platforms HDFS has been designed to be easily portable from one platform to another.

HDFS Implementations[3]

Hardware Failure Redundant copies of the data are kept by the system.

Streaming Data Access Applications that run on HDFS need streaming access to their data sets. Programs read and write data from and to STDIN and STDOUT.

Large Data Sets An HDFS data file is "chuncked" to minimize total program execution time.

Simple Coherency Model HDFS trades off some POSIX requirements for performance, so some operations may behave differently than you expect them to.

Moving Computation is Cheaper than Moving Data map() are copied to the data and the results are copied to the reducer functions.

Portability Across Heterogeneous Hardware and Software Platforms Systems IAW standards gain market share.

HDFS terminology

Some terms:

namenode manages the filesystem namespace. It maintains the filesystem tree and the metadata for all the files and directories in the tree.

client accesses the filesystem on behalf of the user by communicating with the namenode and datanodes. The client presents a POSIX-like filesystem interface.

datanode are the workhorses of the filesystem. They store and retrieve blocks when they are told to (by clients or the namenode).

Our applications are clients, and the mysteries of the name and data nodes are hidden from us.

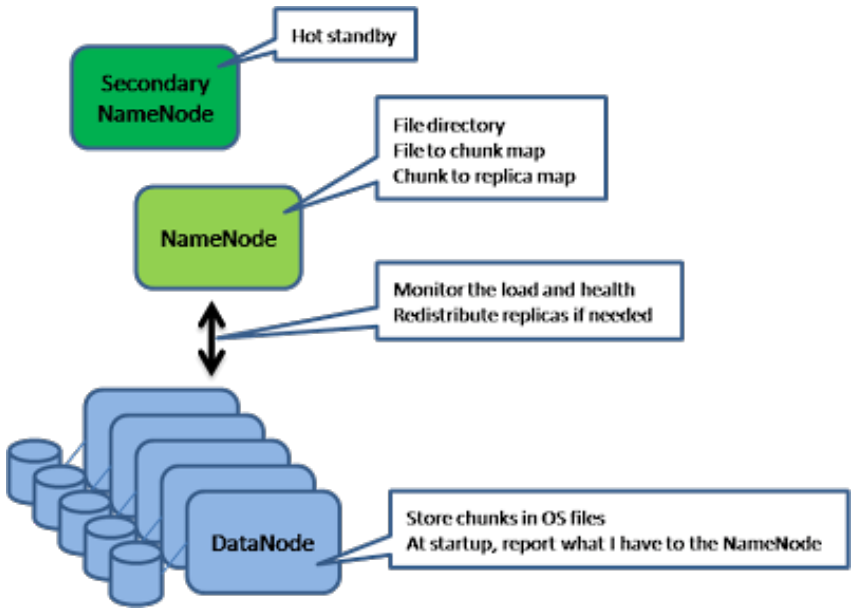


Image from [1].

Same image.

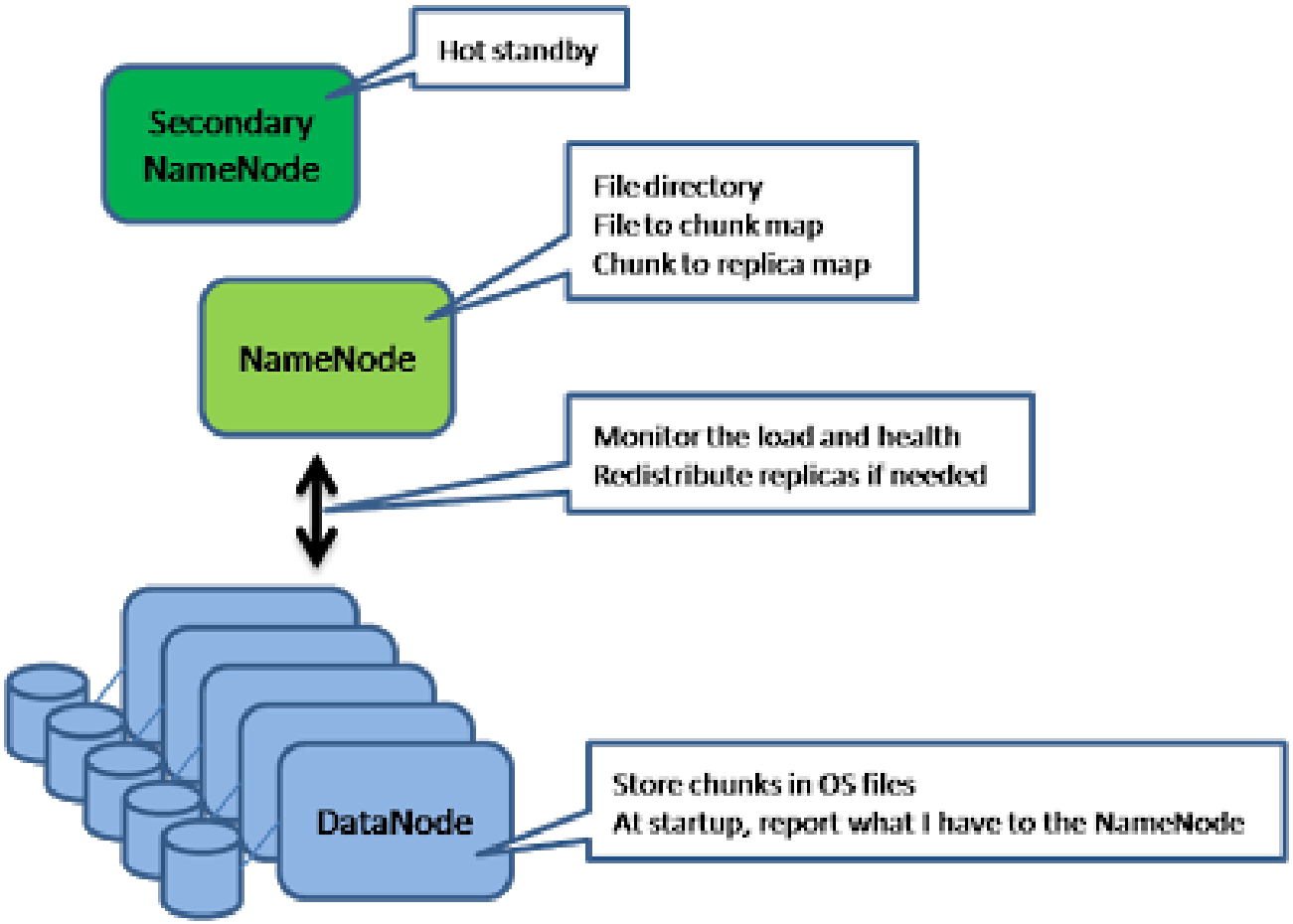


Image from [1].

Map reduce model from 50,000 foot view.

A simple and powerful model:

1. A line of data is presented to a "mapper" function.
2. The "mapper" outputs 0 or more key and value tuples per presented input line
3. Hadoop sorts and merges all keys and values so that there is one key with one or more values
4. The "reducer" processes each key and associated values to the output

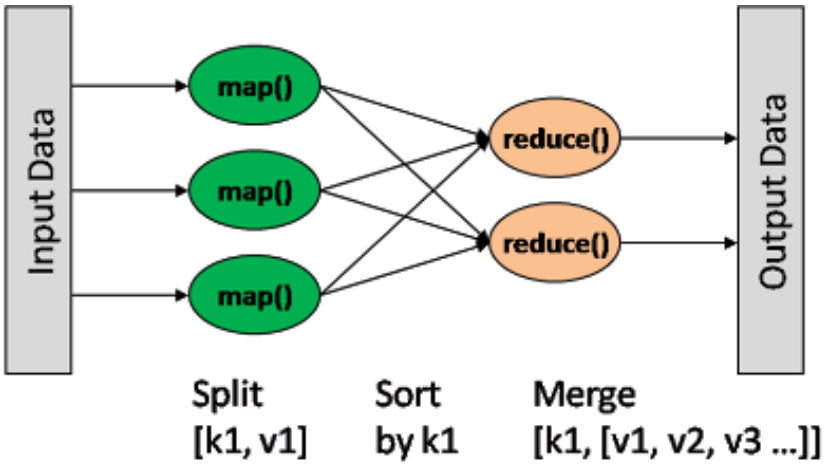


Image from [1].

Same image.

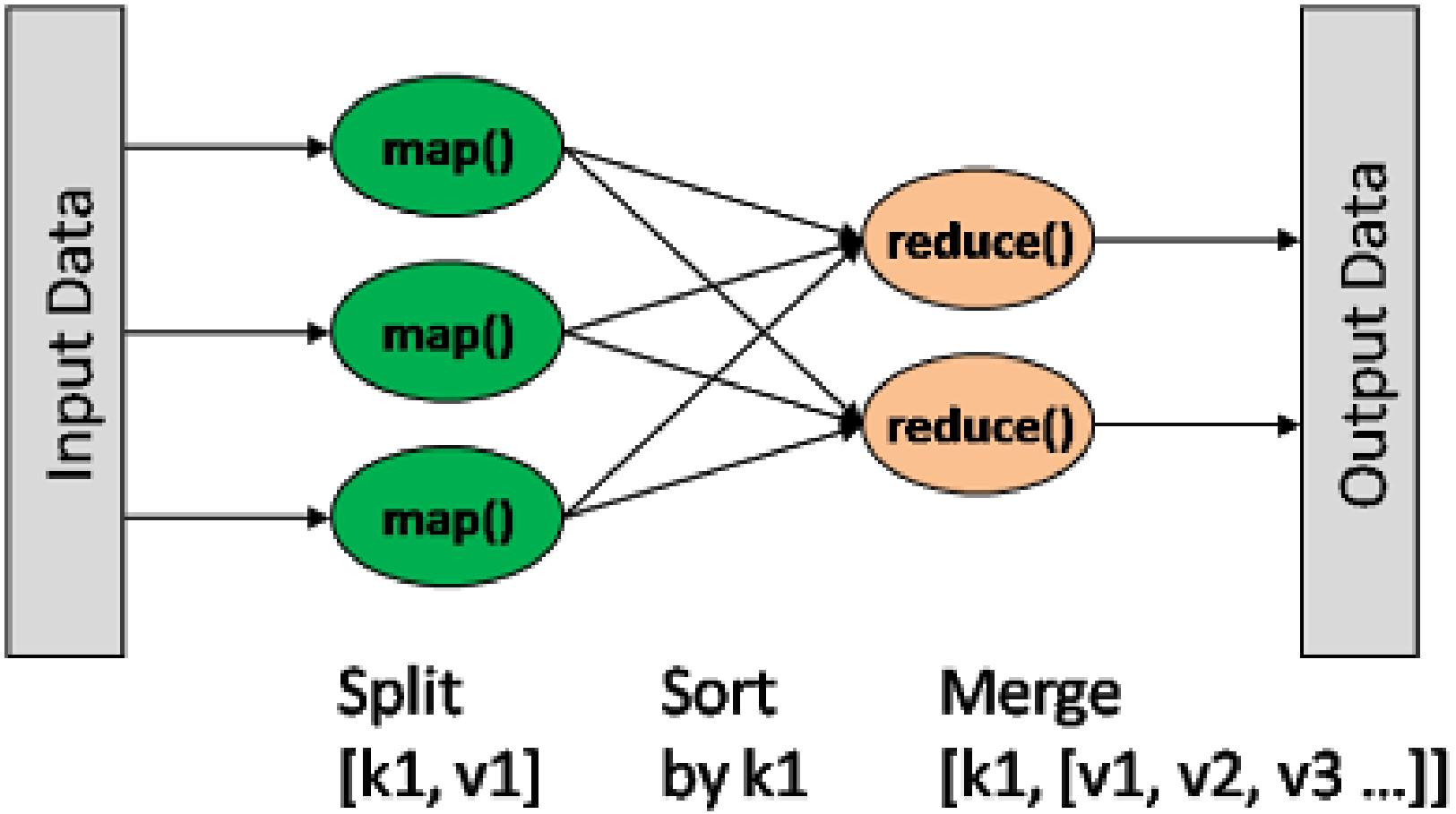


Image from [1].

A lower level view

There are a lot of processes and coordination happening behind the scenes. The client submits a job to Hadoop, mapper functions are copied to the data, key values are sorted, then presented to the reducers, and output is written. Much of this activity can be monitored at port 8787.

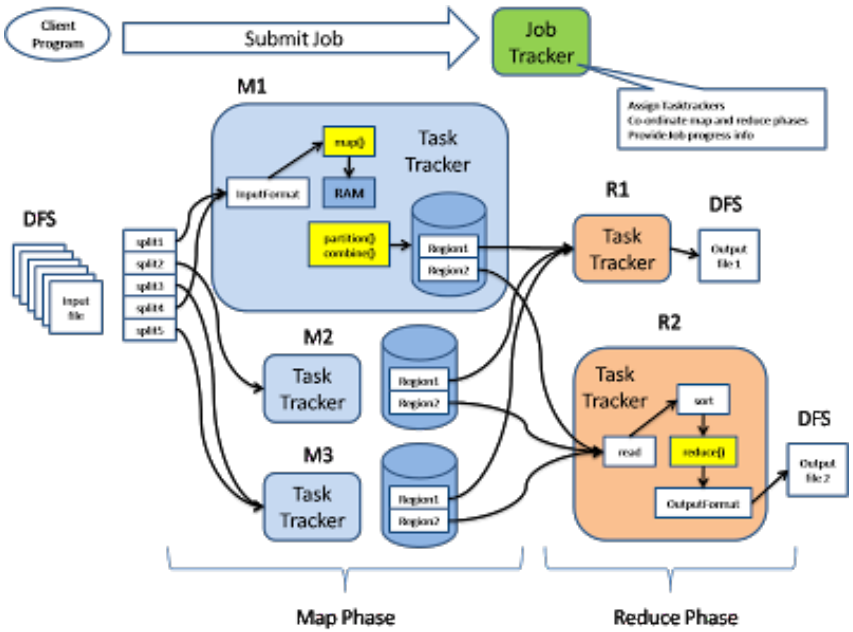


Image from [1].

Same image.

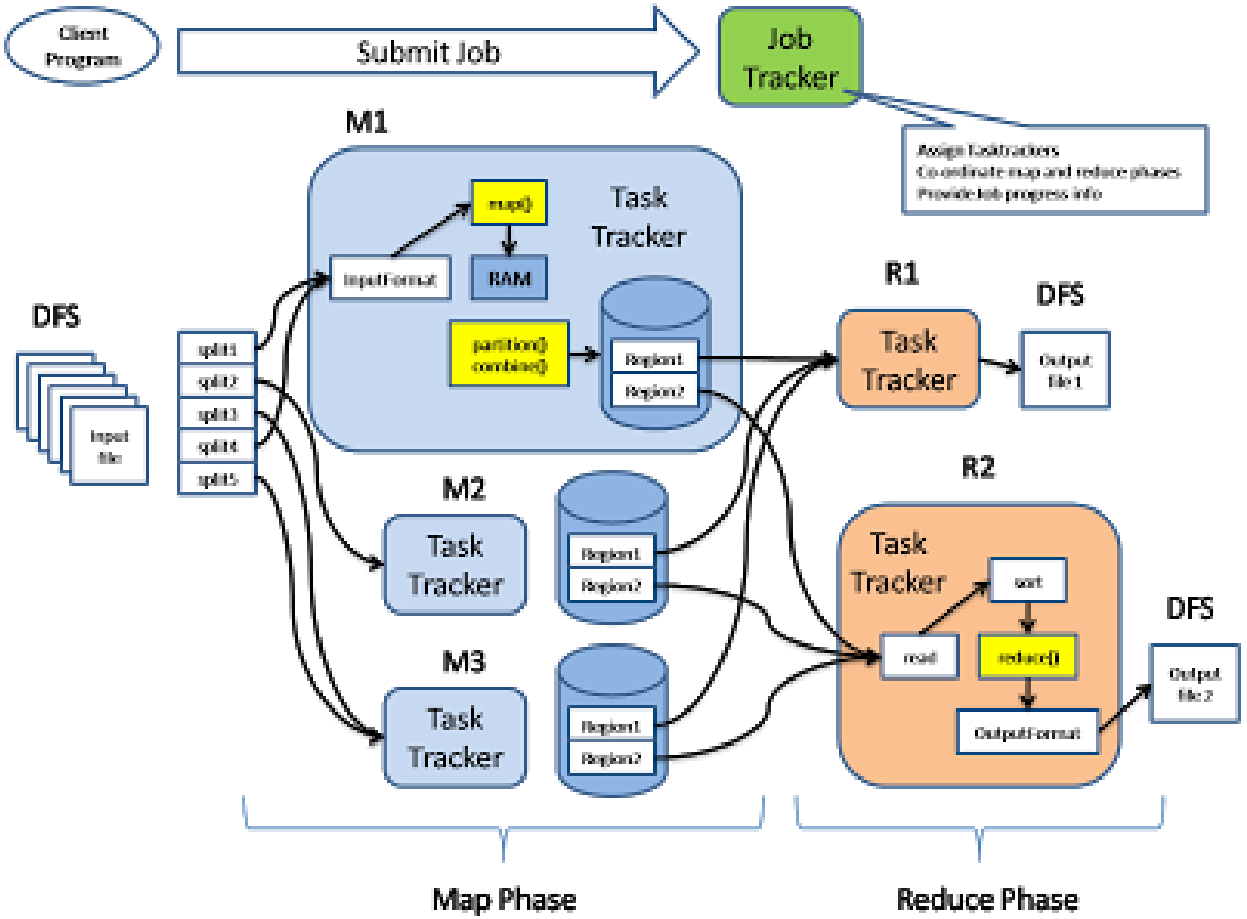
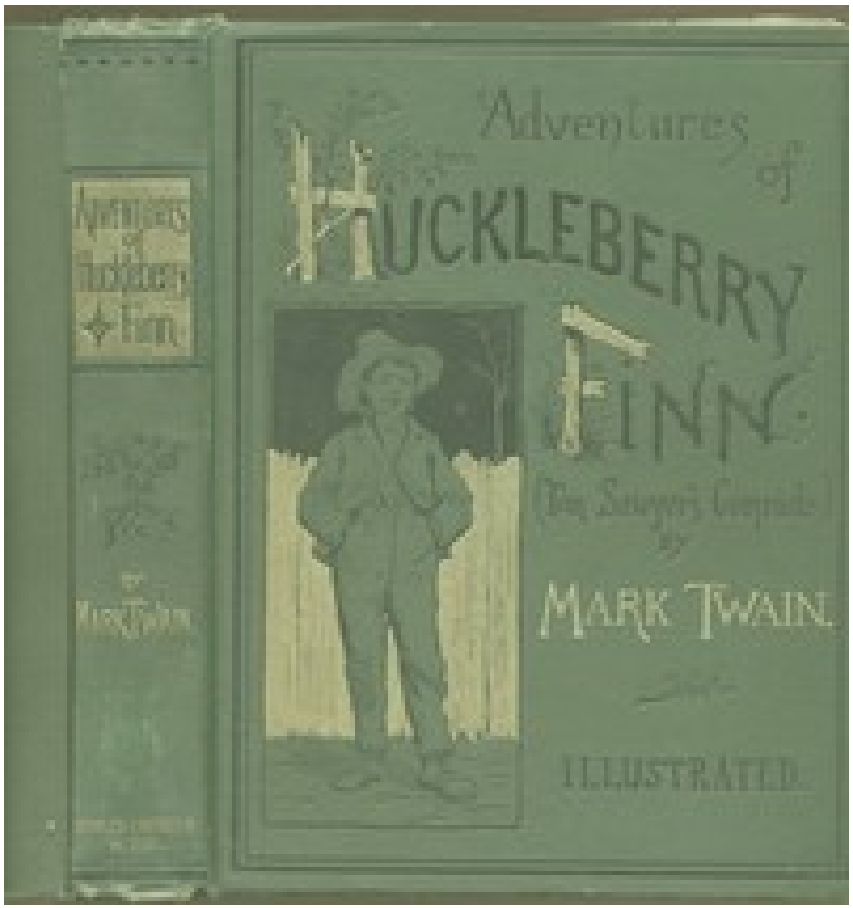


Image from [1].

Classic word count program

The program is in the attached file (Hadoop word count). We'll:

1. Set some environment variables for Hadoop
2. Load necessary R libraries
3. Download and save the text file
4. Do some HDFS housekeeping
5. Define and execute the map-reduce job
6. See where the results ended up



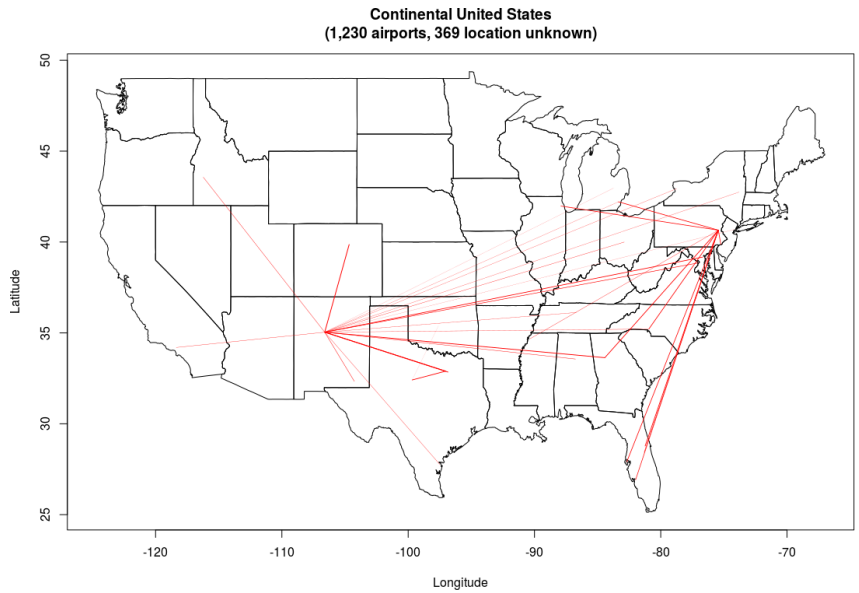
Ways to modify the word count program.

- Remove all "words" that are in fact a space
- Remove all "stop" words
- Remove all words that are numbers
- Stem all words
- Process a different text file
- Create a histogram of the first n most common words
- Estimate the "reading" level of the processed text
- Create a word cloud in the shape of something associated with the text

Looking at air traffic between US domestic airports (attached Airport route exploration)

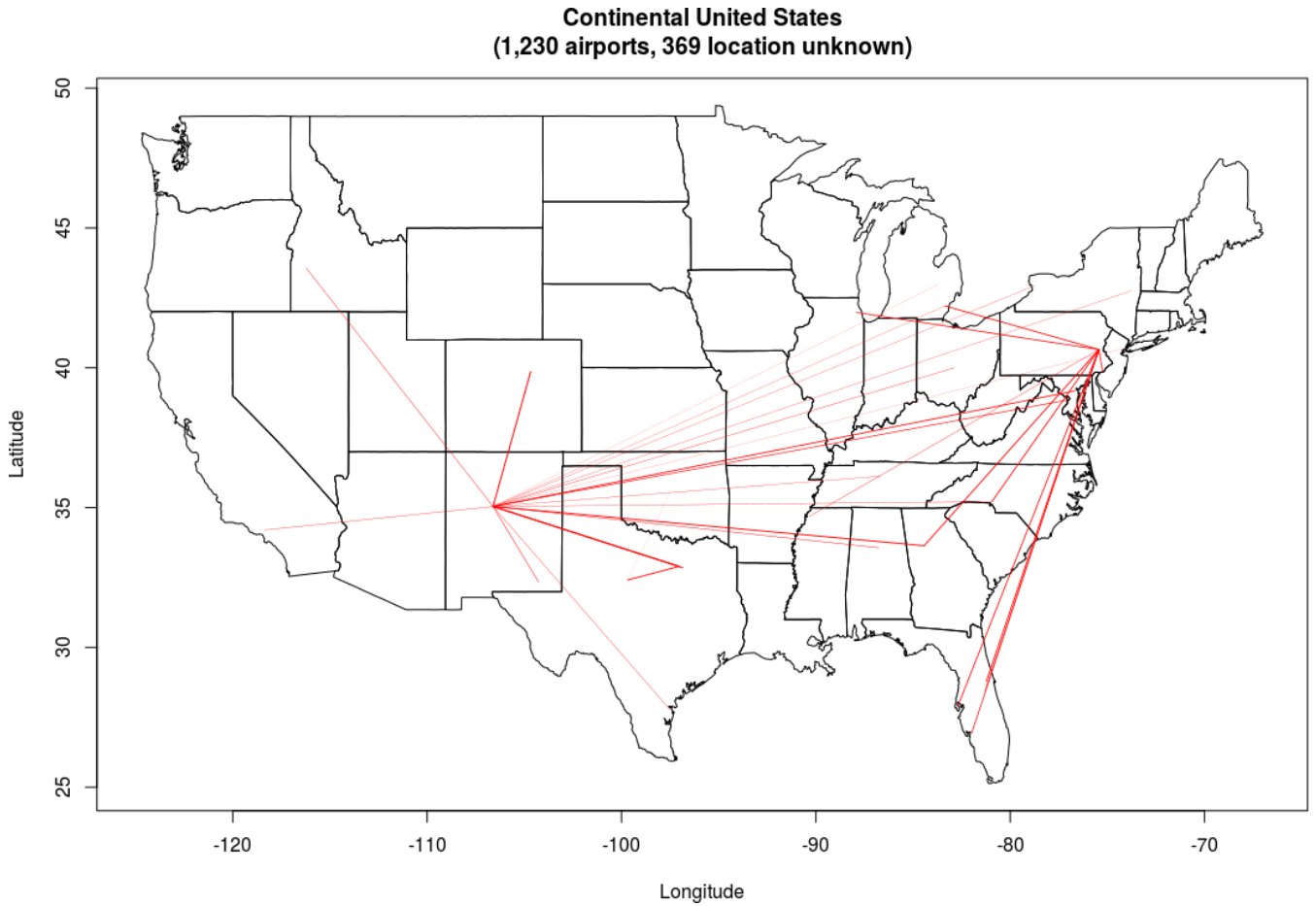
Mashing data from different sources.

- Use the US Government Bureau of Transportation Statistics to get route data
- Use the OpenFlights to find airport latitude and longitude
- Use Hadoop map/reduce model to create a pivot table
- Plot results



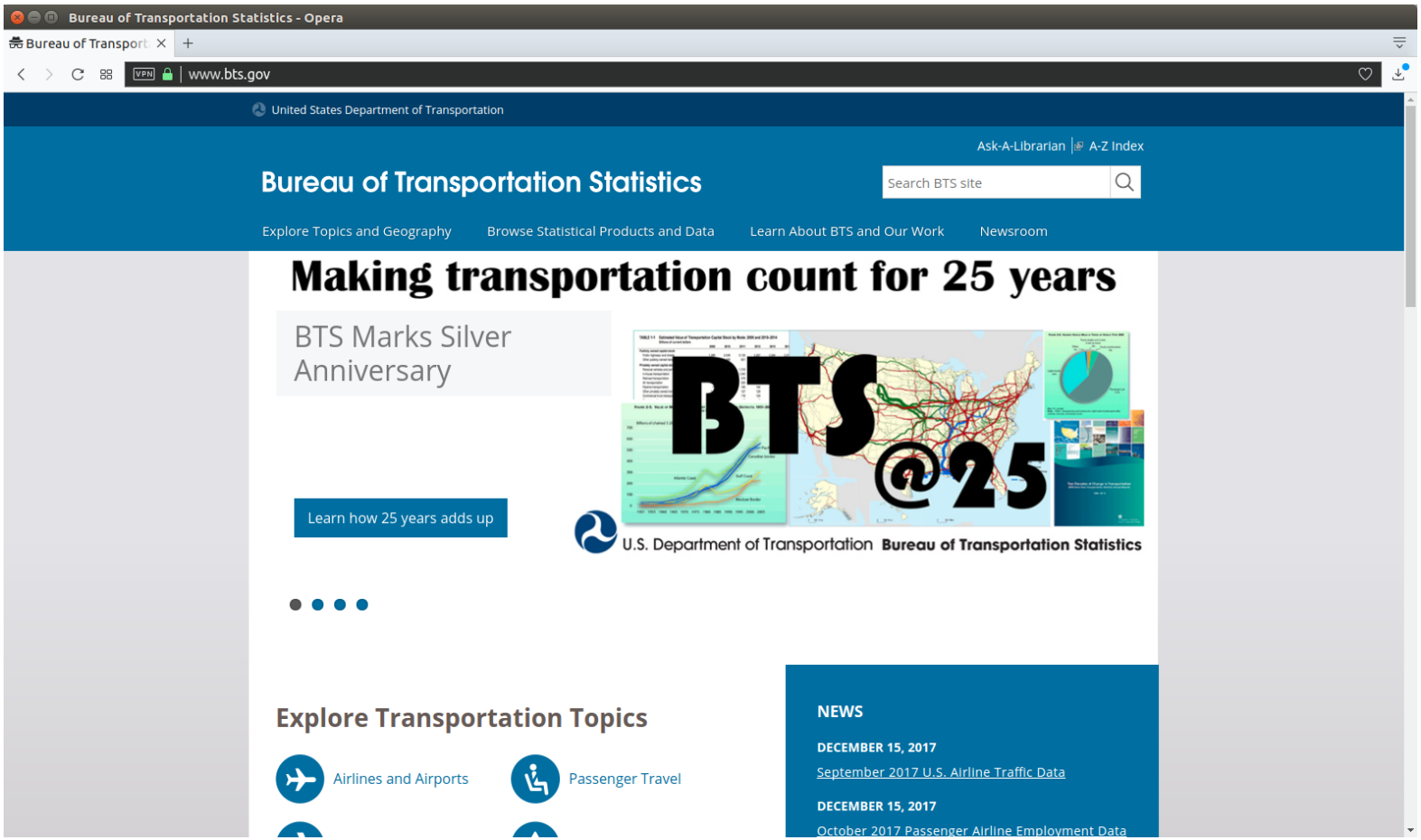
Attached file.

Same image.



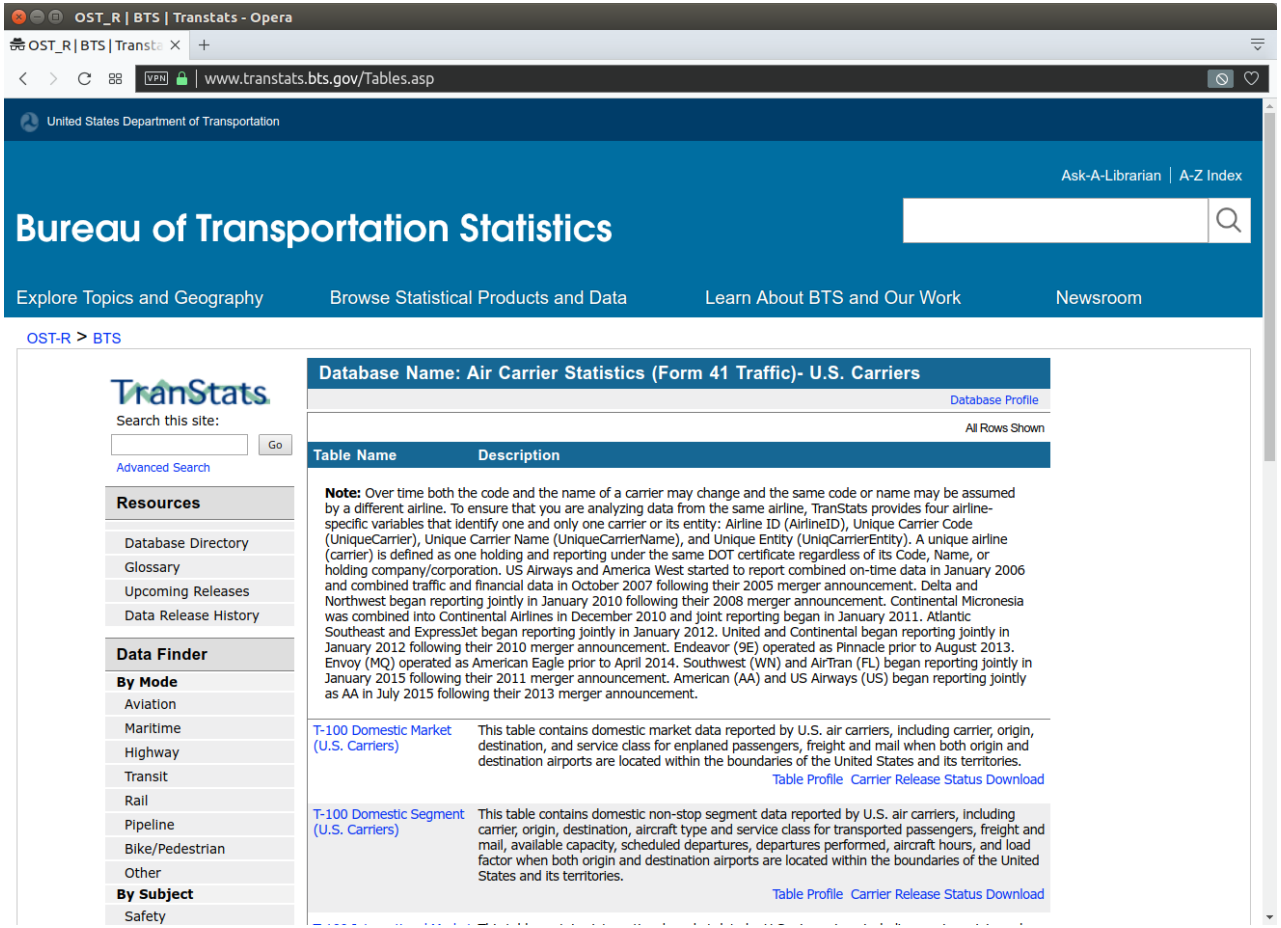
Attached file.

Bureau of Transportation Statistics home page



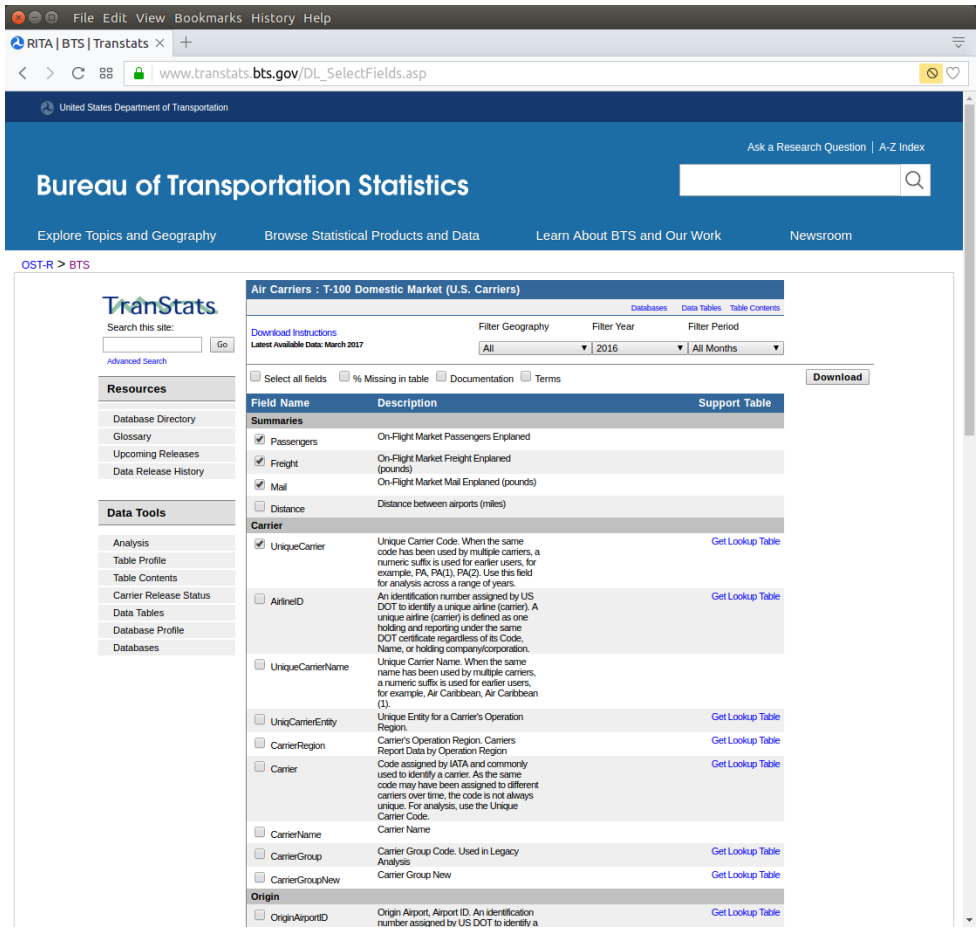
https://www.bts.gov

BTS Airlines and Airports page



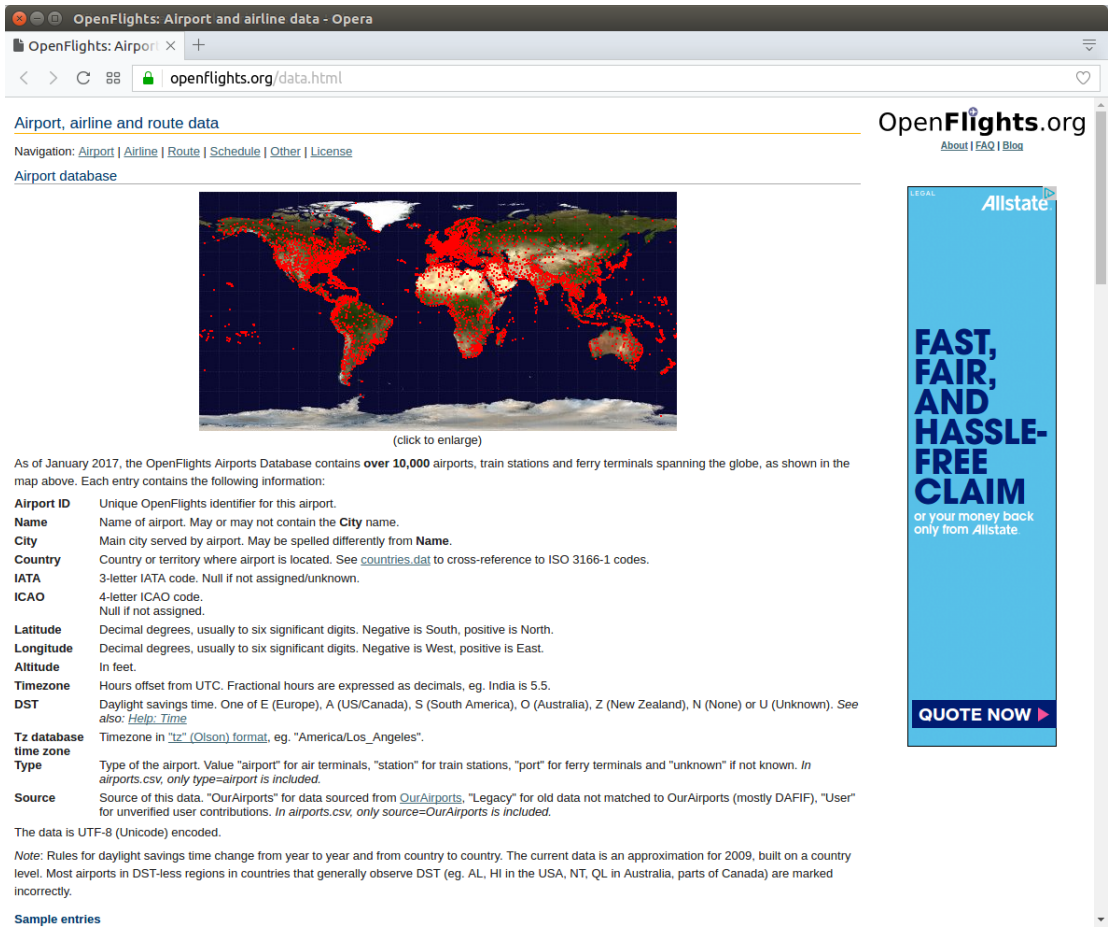
`https://www.transtats.bts.gov/Tables.asp?DB_ID=110&DB_Name=Air%20Carrier%20Statistics%20%28Form%2041%20Traffic%`

BTS Domestic Data Selection page



`https://www.transtats.bts.gov/DL_SelectFields.asp`

OpenFlights home page



https://openflights.org/data.html

Lessons learned about keyval()

Some "interesting" things about the `keyval()` function:

1. The last call wins. If your processing creates a collection of key value pairs, the last `keyval()` call is the data passed to `reduce()`.
2. `keyval()` is vectorized. There can be more than one key or value passed to the function.

To pass more than one key value combination, use:
`keyval(c(...), c(...))`
Be aware that the shorter argument will be recycled as necessary to match the longer argument.
Execute keyval at the R prompt to see code.

Ways to modify the airport program.

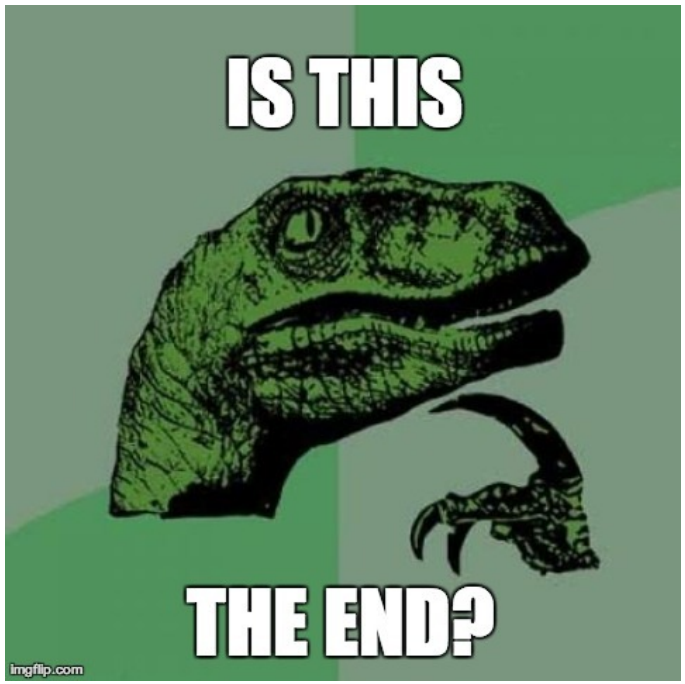
- Change the lines between airports to great circle routes
- Reduce the number of routes to those that carry the greatest weight
- See the difference between cargo and passenger routes
- Modify the routes to show source and destination
- Identify the most common carriers by weight
- Identify the most frequent carriers
- Compute net weight exchange between airports (find sources and sinks)
- If the data is for US domestic routes, why are there links to Chile
- Expand the list of airport locations to remove all unknown locations

Q & A time.

Q: How many Oregonians does it take to screw in a light bulb?
A: Three. One to screw in the light bulb and two to fend off all those Californians trying to share the experience.

What have we covered?

- Gained an understanding of how R interfaces with the Hadoop map-reduce programming model
- "Played" with a word count program
- Looked at things that airlines carry between airports and how to display that data



Next: BDAR Chapter 5, RDBMSs and R

References (1 of 1)

[1] Ricky Ho, How Hadoop Map/Reduce works, 2008.

[2] Apache Staff, HDFS Architecture Guide, `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`, 2017.

[3] Tom White, Hadoop: The Definitive Guide, 4th Edition, O'Reilly Media, Inc., 2015.

Files of interest

1. Hadoop word count
2. Airport route exploration
3. R library script file
4. Route information
5. Presentation reporting the analysis of political party platforms
6. Reporting the analysis of political party platforms
7. Explorations into textual mining