Analysis of Wikipedia Traffic Data in Weka, Hadoop and Spark

COEN242 Big Data
Prof Ezzat
Group 2

Team Members

Pawan Kumbhare <pkumbhare@scu.edu>, W1175795
Manasa Nagaraju <mnagaraju@scu.edu>,W1189995
Guang Shi <gshi@scu.edu>,W1175506
Guoying Wang <gwang3@scu.edu> W1089866
Jessie(Jie) Zheng <jzheng2@scu.edu>, W1071231

Outline

1. Introduction
2. Methodology
3. Implementation
4. Performance Analysis and Discussion
5. Challenges
6. Conclusion

1.INTRODUCTION

1.1 Motivation

- **Interested in real-time online data**
- **Importance of Wiki**
  - one of the most visited web sites; 13,500 million visits every month(Reinoso et al. 2012)
  - Useful information hidden in visiting data
    - predict future events: electoral prediction (Yasseri and Bright, 2016) and stock market trends (Moat et al. 2013).

1.INTRODUCTION

1.2 Objective

- **Questions we want to answer**
  - Overall most visited pages?
  - Hourly most visited pages?
  - Which article is read most during a month?
  - Whether we can predict the project type?
  - Whether we can cluster the data?

1.INTRODUCTION

1.2 Objective

- **Performance analysis and comparison among Weka, Hadoop and Spark**
  - Processing time
  - Data size that can be operated on

2.METHODOLOGY

2.1 Dataset

- 490G+ zipped sample of hourly wikipedia article traffic statistic dataset spanning a 7-month period (January 1 to July 31, 2016)
  - https://dumps.wikimedia.org/other/pagecounts-raw/2016/2016-01/
  - hourly wikipedia article traffic statistics dataset
  - regularly logged from the wikipedia squid proxy by Domas Mituzas
  - Each log file is named with the date and time of collection: pagecounts-20090430-230000.gz
  - Each line has 4 fields: projectcode, pagename, pageviews, bytes

2.METHODOLOGY

2.1 Dataset

```
domain_code page_title count_views total_response_size

en Main_Page 42 50043
```

42 requests to "en.wikipedia.org/wiki/Main_Page", which accounted in total for 50043 response bytes

```
en Barack_Obama 997 123091092
en Barack_Obama%27s_first_100_days 8 850127
en Barack_Obama,_Jr 1 144103
en Barack_Obama,_Sr. 37 938821
en Barack_Obama_%22HOPE%22_poster 4 81005
en Barack_Obama_%22Hope%22_poster 5 102081
```

2.METHODOLOGY

Domain trailing part	Coded as	Database name
`.wikipedia.org`		`*wiki` (be careful about the other non- wikipedia sites using this however)
`.wikibooks.org`	`.b`	`*wikibooks`
`.wiktionary.org`	`.d`	`*wiktionary`

```
de.m.voy Berlin 176 314159
```

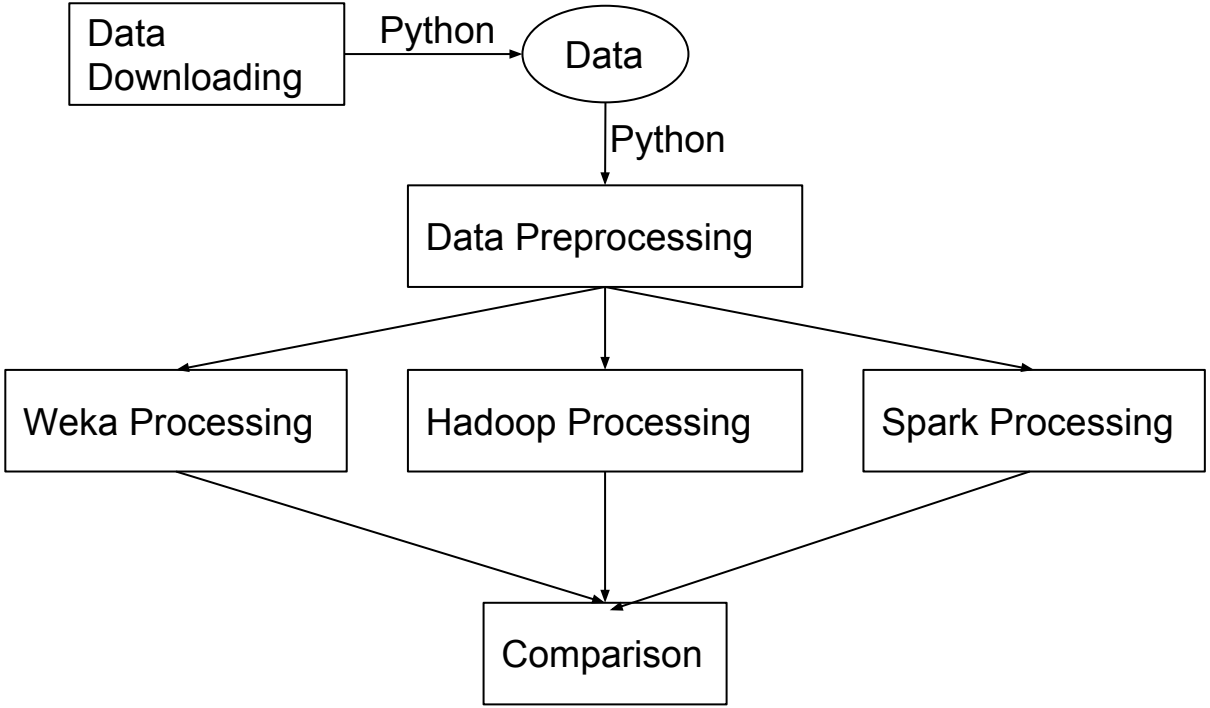
176 requests to "de.m.wikivoyage.org/wiki/Berlin", which accounted in total for 314159 response bytes

2.METHODOLOGY

2.2 Algorithms

- Classification Analysis:
  - Naive Bayesian Algorithm: To classify new topic into category
- Cluster Analysis
  - K-means Algorithm: To identify the categories

2.METHODOLOGY



Data Preprocessing

Data Preprocessing

- **Data size is large**
  - 70G+ zipped text file per month
  - 7 months' data
  - After unzip we have over TERABYTE raw text file to be processed.
- **Data Cleaning**
  - Missing value: very few
  - Invalid value:
    - Missing features
    - Unmatched format
- **Feature Extraction**
  - Use all features

Data Preprocessing

- **Implementation in Python**
  - ***Step 1***: Downloading the data
    - 2mins for one hour data; **7 days** for 7 month data
  - ***Step 2:*** Preprocessing the data and get rid of low/zero visit entries.
    - Extract top 500 pages with highest view counts for each language
    - 7mins for one hour data; **24.5 days** for 7 month data

Data Preprocessing

- **Implementation in Python (cont.)**
  - ***Step 3***: First level of cleaning
    - Removing entries with missing features
    - Updating these entries manually
  - ***Step 4***: Second level of cleaning
    - Removing entries in which the data format is mismatching
      - E.g. additional characters in numerical field
    - Updating these entries manually

Data Preprocessing

- **Outlier**
  - Domain code = page title (e.g. en = en)
- **Data Transformation**
  - Usually we need to normalize the data before clustering
  - Weka, Mahout and Spark normalize data internally

Weka

Weka specific preprocessing

- **Loading the data.**
  - 3GB csv will occupy more than 10GB memory for loading.
  - Resampling.
  - Class balancer.
- **Data cleaning**
  - Finding and modifying errors for huge file.
  - Identifying unknown errors.

Naive Bayes Analysis

- 416355 entry has exercised. With 66% being the training set and the rest being test set.
- Naive Bayes algorithm is giving out 77.6% prediction accuracy.
- 38MB data took 8.57GB memory to do the needed processing.

Naive Bayes Results

```
=== Confusion Matrix ===

   a    b    c    d    e    f    g    h    i    j    k    l   <-- classified as
4886    0    0    0    3    5    0    0    0    0    0    0 |    a = m
   0 1191  589    0    3    0    0    0    0    0   51    0 |    b = wd
   0   56  693    0    1    1    0    0    0    0   15    0 |    c = w
   3    0    0 4049    0   13    0  114   51   38    0    5 |    d = d
  13   15   60  330  125   99   21   55   75   59   42    0 |    e = mw
   0    0    0  608    6  236    0   64   59   34    1    0 |    f = b
   0    2    0   11    1    0  129    4    1    3    0    0 |    g = v
   0    0   64  459    0    5    0 1148   35   27   14    3 |    h = s
   0    0    0  314    0    0    0   66  175   21    0    1 |    i = q
   0    0    0  140    0    4    0   24    8  470    0    0 |    j = n
   0    8   85    0    0    0    0    0    0    0  182    0 |    k = f
   0    0    0   26    1    5    0   10    2    4    0   42 |    l = voy
```

Naive Bayes vs J48 decision tree

- **Naive Bayes**
  - 1.66G memory needed to process 7.7MB data
  - Time taken to build model: 0.07 seconds
  - Time taken to test model on training split: 0.83 seconds
  - Correctly Classified Instances 76.5706 %
- **J48 decision tree**
  - 7GB memory needed to process 7.7MB data
  - Time taken to build model: 8.9 seconds
  - Time taken to test model on training split: 0.41 seconds
  - Correctly Classified Instances 77.6657 %

K means performance vs Cluster counts

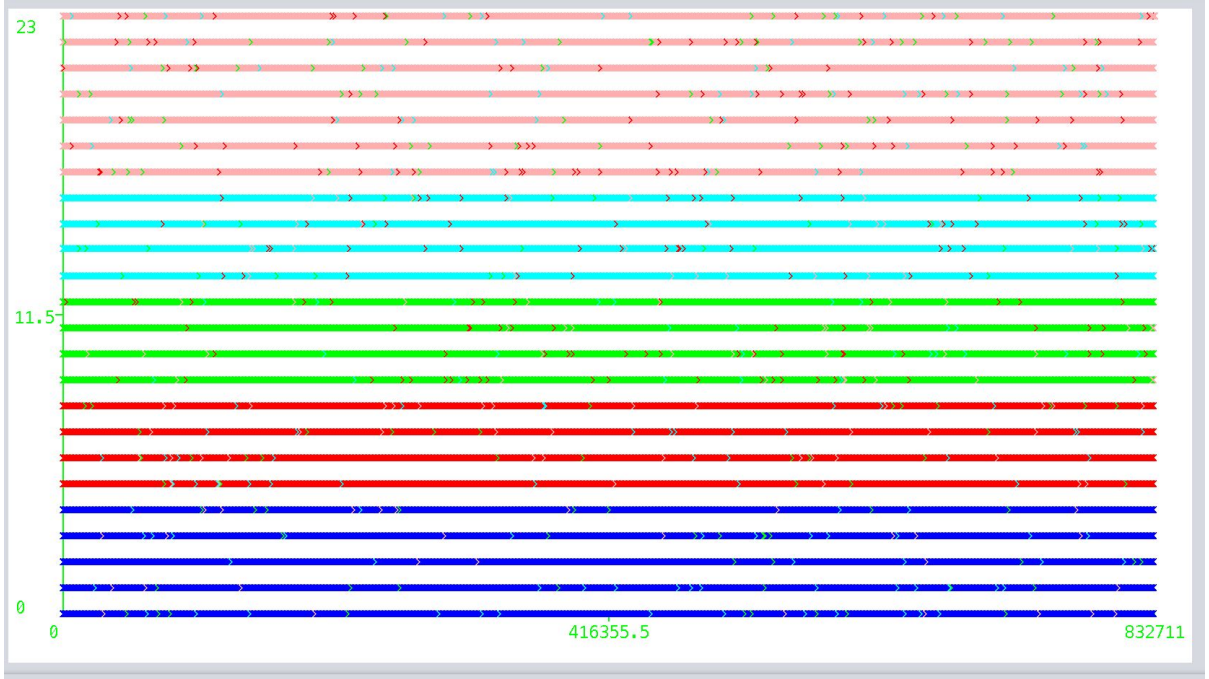
Using hour, language and visit count attributes to cluster, Comparing the time it takes to finish.

Data size: 77MB

K	2	4	8	16	32	64
Time(s)	1.04	1.49	2.6	5.88	11.82	22.26

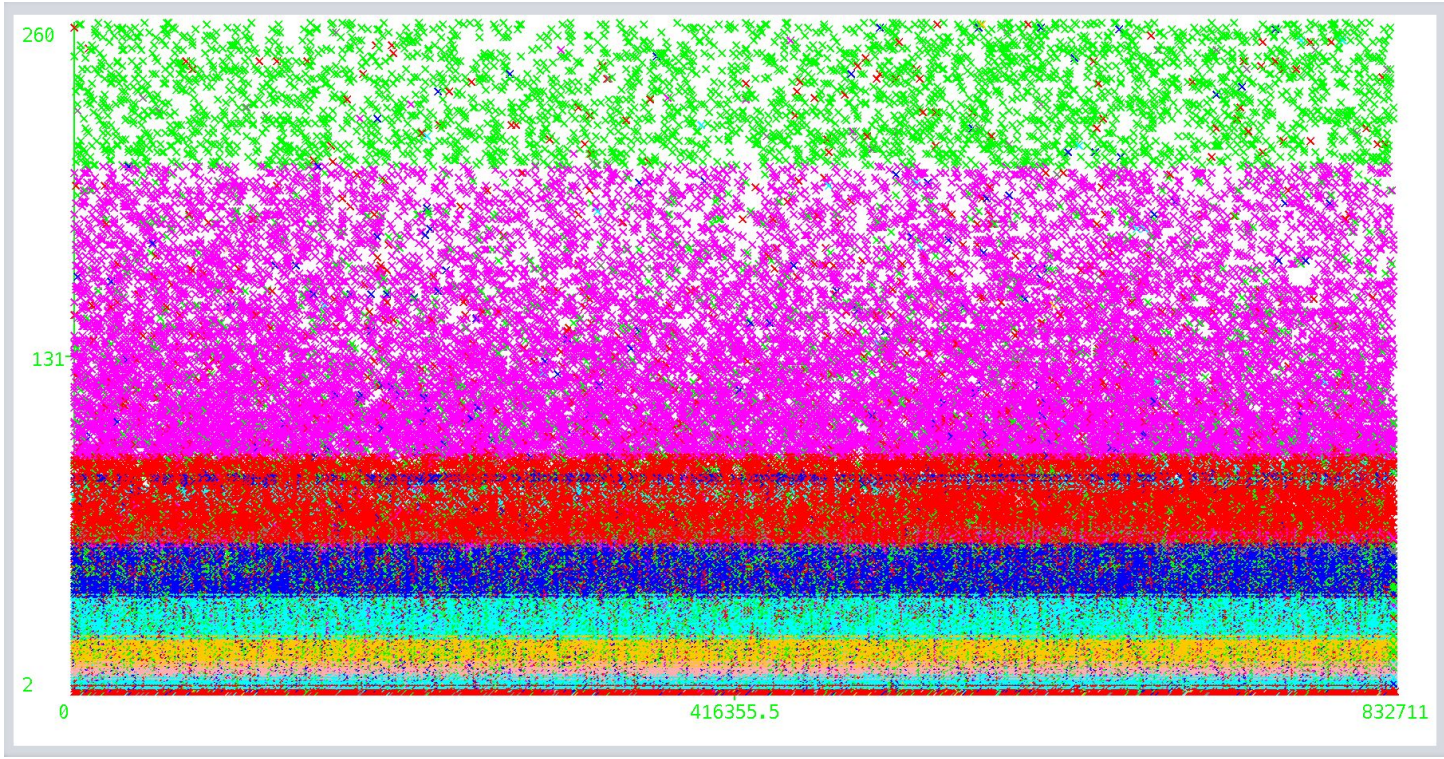
Kmeans Clustering

Using hour, language and visit count attributes to cluster, Plot hourly distribution.



Kmeans Clustering

Using language, category and visit count attributes to cluster, Plot visit count distribution.



Mahout/Hadoop

Mahout MapReduce

- Apache Mahout is an open source project primarily used for creating scalable machine learning algorithms
- Mahout MapReduce supports Clustering (K-means, Canopy, ...), Classification (Naive Bayes, Logistic Regression,...), Recommendation.
- Runs on HDFS

Mahout Execution Steps:

1. Preprocessing:

- Generate Sequence File [Key/Value] file
  - mahout seqdirectory -c UTF-8 -i ./wiki-input -o ./wiki-seqfiles
    (generates sequence file with filename as key and content as value [Not useful for our analysis])
  - Implemented CSV to Sequence File generation file [wikiCSVDir2Seq.java] → wikiSeqOutput
    hadoop jar CSVDir2Seq.jar wikiCSVDirToSeq input wikiSeqOutput

- Convert Sequence Files to Vectors
  mahout seq2sparse -i wikiSeqOutput/ -o wikiVectors/ -ow -chunk 100 -x 90 -seq -ml 50 -n 2 -nv

Mahout Execution Steps:

2. Mahout Naive Bayes Execution

Mahout splits the set into two sets: a training set and a testing set

2.1 Training model

Using the training set to train the model

$mahout split -i naiveVectors/tfidf-vectors --trainingOutput train-vectors --testOutput test-vectors --randomSelectionPct 40 --overwrite --sequenceFiles -xm sequential

$ mahout trainnb -i trainvectors -el -li labelindex -o model -ow -c

2.2 Testing the data

Testing the model on both training data set and testing data set

$mahout testnb -i trainvectors -m model -l labelindex -ow -o testingdata -c

Mahout ---- Naive Bayes Execution

```
16/08/21 00:36:47 INFO test.TestNaiveBayesDriver: Complementary Results:
=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :      12049      65.4481%
Incorrectly Classified Instances        :       6361      34.5519%
Total Classified Instances              :      18410

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       e       f       g       h       i       j       k       l       <--Classified as
788     82      53      17      42      192     242     156     52      81      22      22       |  1749        a     = b
111     713     95      35      101     146     174     102     93      111     47      65       |  1793        b     = d
5       16      1554    22      7       17      9       16      10      7       52      21       |  1736        c     = f
28      37      25      1526    16      19      16      13      21      11      30      28       |  1770        d     = m
67      78      70      40      1141    15      104     52      25      68      17      40       |  1717        e     = mw
12      19      38      6       21      1351    74      66      22      70      7       14       |  1700        f     = n
43      97      73      15      38      329     747     182     55      126     26      27       |  1758        g     = q
23      38      38      8       19      261     116     1077    38      45      20      22       |  1705        h     = s
16      28      24      19      7       77      23      117     1316    81      26      25       |  1759        i     = v
15      5       22      12      48      357     14      129     32      1107    14      14       |  1769        j     = voy
8       11      102     15      5       5       6       4       14      2       729     53       |  954         k     = w
0       0       0       0       0       0       0       0       0       0       0       0        |  0           l     = wd

=======================================================
Statistics
-------------------------------------------------------
Kappa                                       0.6155
Accuracy                                   65.4481%
Reliability                                55.8416%
Reliability (standard deviation)            0.2944

16/08/21 00:36:47 INFO driver.MahoutDriver: Program took 15066 ms (Minutes: 0.2511)
[bigdata02@linux60812 guoying]$
```

Use project_code as classifier, together with all other attributes

(wikibooks: ".b" ; wiktionary: ".d" ; wikimedia: ".m" ; wikipedia mobile: ".mw" ; wikinews: ".n" …….)

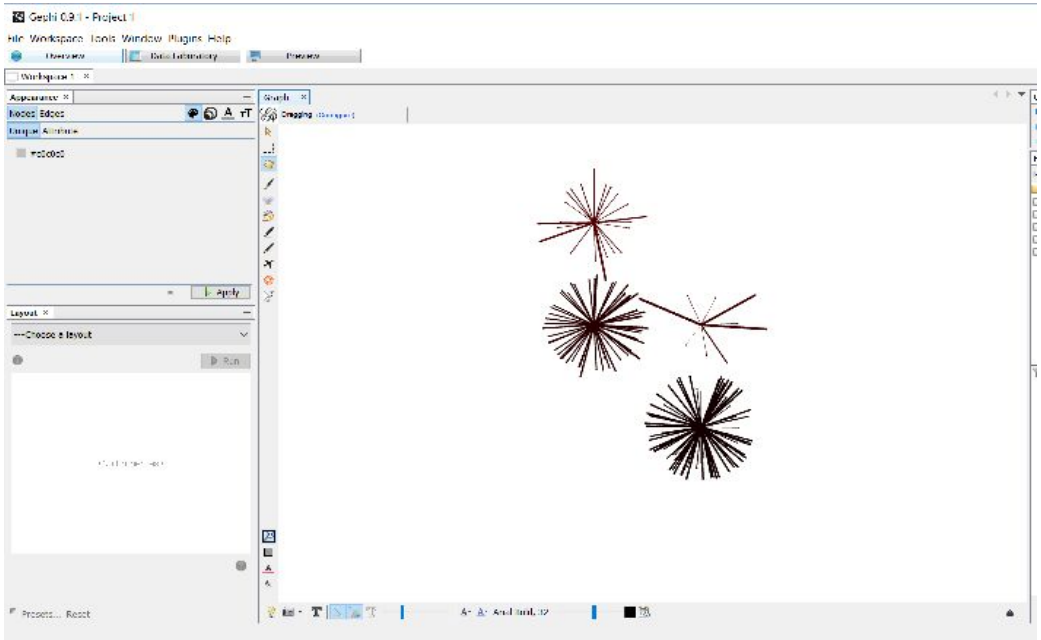
Mahout Execution Steps:

3. Mahout K-means Clustering Execution

- **Specify cluster parameters**
  - $ mahout kmeans -i wikiVectors/tfidf-vectors/ -c wiki-kmeans-centroids -cl -o wiki-kmeans-clusters -k 3 -ow -x 10 -dm org.apache.mahout.common.distance.CosineDistanceMeasure
- **Produce the result:**
  - $ mahout clusterdump -d wikiVectors/dictionary.file-0 -dt sequencefile -i wiki-kmeans-clusters/clusters-2-final -n 3 -b 100 -o cdump.txt -p wiki-kmeans-clusters/clusteredPoints/
- **Visualize in Gephi**
  - $ mahout clusterdump -i wiki-kmeans-clusters/*-final -p wiki-kmeans-clusters/clusteredPoints/ -of GRAPH_ML -o graphml

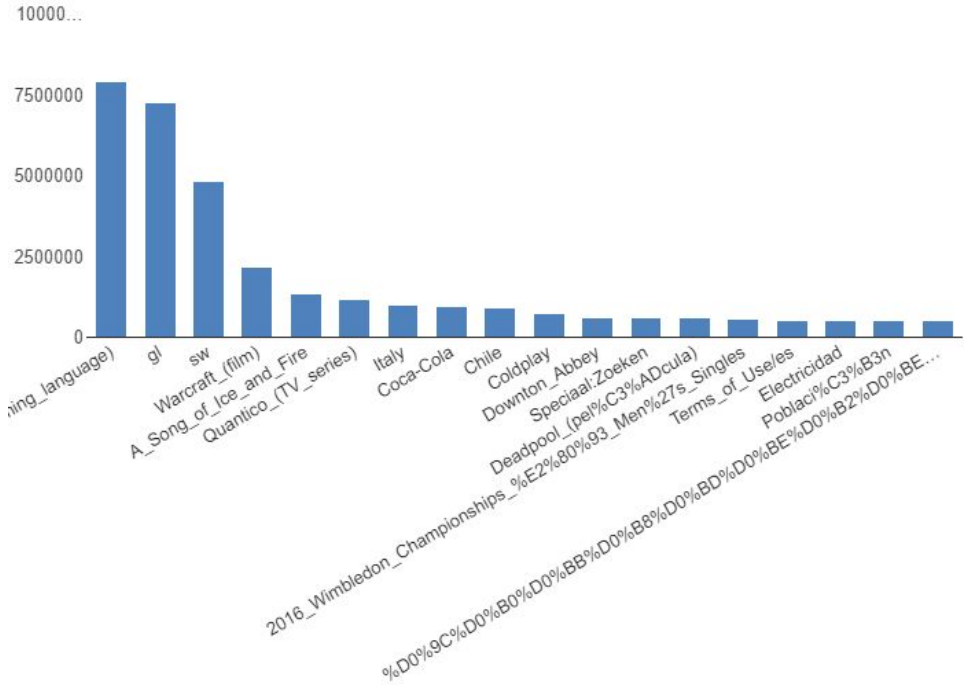
K Means Clustering

Using language, pageviews and hour as attributes



K	Time(min)
2	1.0498166
4	1.00985
8	1.0208833
16	0.6256
32	0.6527
64	0.6509

MapReduce function in Hadoop



Implementing Map Reduce to find the top pages having maximum page view

Project name vs Page Views

Time Taken : 68 sec

Spark

Introduction

The key features of Spark include the following:

- Easy to use
- Fast
- General-purpose
- Scalable
- Fault tolerant

Converting our dataset files to RDD

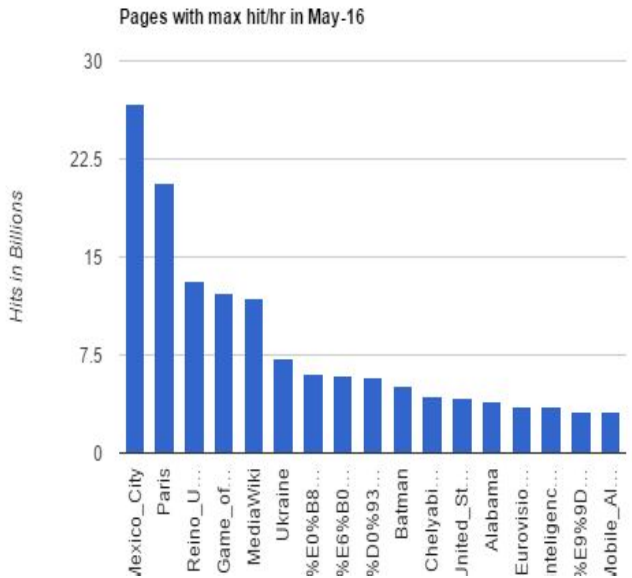
Reading all the input csv files from HDFS and converting them to RDD -

```
val wikiText = sc.textFile("/user/bigdata02/pawan/wiki_*.csv")

case class Wiki(date: String, hour: String, lang: String, proj_type: String, project: String,
    count: Double, bytes: Double)

val wiki = wikiText.map(_.split("`")).map(w =>
    try{Wiki(w(0),w(1),w(2),w(3),w(4),w(5).toDouble,w(6).replaceAll("[\\\\n\"]",
    "").toDouble)})
```

MapReduce function in Spark



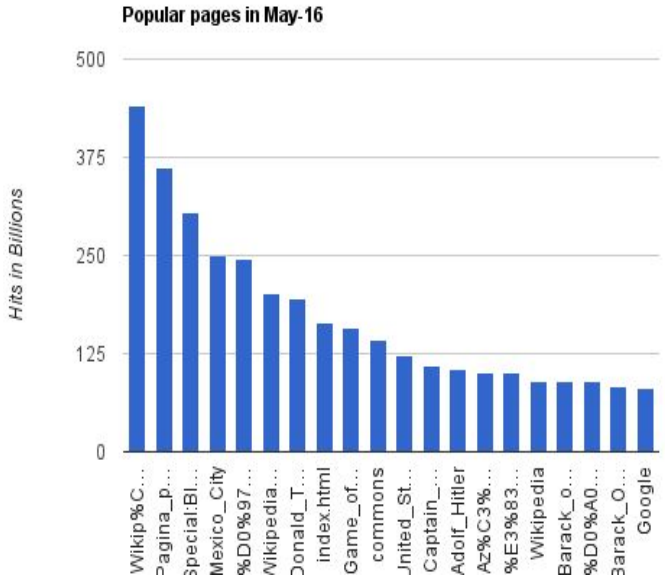
Implementing MapReduce function in Spark to find top pages having maximum hit/hr -

wiki.map(w => (w.project, w.bytes)).reduceByKey(_ max _).map(w => (w._1, w._2)).sortBy(-_._2).take(100).foreach(println)

Data Size = 17.5 Gb

Time Taken to process : ~250,000,000 rows => ~16 sec :)

MapReduce function in Spark cont..



Implementing MapReduce function in Spark to find top pages having overall maximum hits -

wiki.map(w => (w.project, w.bytes)).reduceByKey(_ + _).map(w => (w._1, w._2)).sortBy(-_._2).take(100).foreach(println)

Data Size = 17.5 Gb

Time Taken to process : ~250,000,000 rows => ~16 sec :)

Naive Bayes Classification

Divided the hit counts into three categories -

[Most Popular (3), Popular (2), Not Much Popular (1)]

Feature used = Hits / hour

Training Data = 70%, Testing Data = 30%

Accuracy while testing = 98.76%

Total time taken is 56s

Top Pages in 2016 (Jan-July)

Comparison Summary

Clustering speed comparison:

- For Hadoop:
  - *17.5 G takes 1.11m, Processing speed is (1.11* 60) / (17*1024) = 0.0038s/Mb*
  - *77MB takes 1.0112m, Processing speed is (1.0112*60)/77 = 0.78s/Mb*
- For Weka:
  - *77MB takes 1.73s, Processing speed is 1.73/77 = 0.0225s/Mb*

MapReduce speed comparison:

- For Hadoop:
  - *17.5 G containing ~250,000,000 rows takes around 68s*
- For Spark:
  - *17.5 G containing ~250,000,000 rows takes around 16s*

Future Scope

- After fetching the categories of each wiki project, we can get popular categories that users are interested in.

- To do time-series analysis on the dataset.

DEMO