POPULARITY ANALYSIS OF WIKIPEDIA ARTICLES

-Rafica Abdul Rahim (ra2688)

Abstract:

The main focus of this project is on Information Retrieval and data analysis of large data sets. In this project, I am building an application which ranks the English Wikipedia articles based on the page views and popularity trends. This application is developed in the Map Reduce programming paradigm. It will use Elastic Map Reduce and Hive cluster service of Amazon Web Services. Every day's hourly page count files for 15 days, will be used as the input dataset, to find the latest trending and popular Wikipedia article. This development process involves, preprocessing of the input, filtering and transforming the input dataset, aggregating the page views, finding popularities of the articles and loading it into hive database to be available for querying.

Introduction:

Data generation is increasing very heavily in many companies. Meaningful information lie under these large data sets and hence information retrieval on these large datasets are important. Advanced technologies are emerging to handle 'Big data', to store and process it efficiently in distributed environment like Hadoop. There are many cases where big data analysis is useful like analyzing the crowd behavior from the web traffic information. Analyzing the crowd behavior on web in websites like Wikipedia, Amazon, Google etc. is very important to understand what people are interested in or why some kind of behavior is observed. This project analyses the Wiki traffic information to find out which articles are popular and receiving higher hits in specific period of time.

Technologies Used:

Selection of the right technologies is important in handling big data. In this project, Hadoop is used which is an open source software framework for storage and large scale processing of data-sets. HDFS is a distributed file system of Hadoop that stores data on the commodity machines providing very high aggregate bandwidth across the cluster. The programming model used is MapReduce which is used for processing large datasets with a parallel distributed algorithm on a cluster. The Map reduce code is developed using Hadoop streaming in Python.

For storing the processed information, Hive is used, which is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query and analysis of large datasets stored in Amazon S3 filesystem. Other possibilities taken into account apart from Hive are Hbase and Amazon Dynamo DB. Among these, hive was selected because of its ease in loading elastic map reduce output and it writing queries which are like SQL, called hive query language (HiveQL).

Implementation:

Input dataset:

The dataset contains traffic information of the wiki artilces published by Wikipedia and it can be found at http://dumps.wikimedia.org/other/pagecounts-raw/2013/2013-06/ . They are also uploaded in the S3 filesystem s3://15719f13wikitraffic/. The size of the dataset was around 110 GB. The considered dataset contains one file for each hour of the 15-day long period for June 2013. Each line of each file contains four fields Projectcode, Pagename, Pageviews, Bytes

Environment:

Map reduce program was deployed in an Amazon EC2 Hadoop streaming cluster which had 1 c1.xlarge master instance, 5 c1.xlarge core instances. Master instances assign tasks to core instances and monitor their status. Core instance peforms task given by Hadoop and store it in Hadoop distributed file system. Running time for the map reduce program for the entire dataset was around 15 minutes.

Hive cluster is launched from Amazon web services. Cluster consists of 1 c1.xlarge instance and 5 c1.xlarge core instances similar to map reduce program deployment environment.
Hive tables are created in this hive cluster from the map reduce output stored in s3.

Map-Reduce Phase:

"Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. The worker node processes the smaller problem, and passes the answer back to its master node.
Mapper1.py has the mapper function which does the preprocessing of the data.

- English Articles filtered: All the articles with project code starting with “En” are English Wikipedia articles are selected and rest are excluded.
- File-format check : Articles ending with image or text-file extension are excluded (.gif, .jpg, .png, .ico, .txt)
- Articles starting with lowercase are filtered out as Wikipedia policy states that all English articles must start with an uppercase character.
- Boilerplate pages : Article names containing 404_error, Main_page, Hypetext_Transfer_protocol, Favicon.ico and Search are excluded
- Some of the characters in the article name is percent encoded. %22 in pagename field is replaced with underscore.
- Titles starting with Media, Special, Talk, File etc. are excluded as these categories need not be considered when finding trending topics.

Date of the pageview is extracted from the filename. Date, pageviews and article name are passed as input to the Reducer1.py
Here the Key is “Articlename-Date-Pageviews”. Map function runs exactly once for each key value and the output of map function is sorted based on key values.

"Reduce" step:
The master node then collects the answers to all the sub-problems and combines them to form the output . In between map and reduce steps, Hadoop framework internally performs operation like partitioning, combining, shuffling of the key value pairs.

In reduce function of reducer1.py, the following steps are carried out.
Extraction:
The article name, pageviews and date are extracted from the input and converted to appropriate datatypes for further calculation.

Processing:
The total page views of each date is found for a particle article. A list of the dates in which the page is viewed and a list of its corresponding number of page views on that date is also maintained. The trend factor for a particular article is calculated by this formula:
Trendfactor = Number of views from days 8 to 15 – number of views from days 1 to 7

Articles with less than 10 total page views are filtered out as they might not be significant.

The number of views in the 15th day, first week and last week is also found and passed as output

The output is of the following format
ArticleName}Date_list}Views_list}Total_views}Trend_factor}first_week}last_week}last_day

The output is stored in an S3 bucket s3://wikitrendanalysis/output/FullRunFinal/

Hive Phase:

Once map reduce output is generated, it is loaded to Hive to retrieve information. The following queries were executed in the master node of the hive cluster.

```
CREATE TABLE wiki_pop(name STRING,date_list STRING,
views_list STRING,
total_views BIGINT,
first_week BIGINT,
last_week BIGINT,
last_day BIGINT,
trend DOUBLE)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '}'
LOCATION 's3://wikitrendanalysis/output/FullRunFinal/';
```

TOP 100 MOST POPULAR ARTICLES IN THE TIME PERIOD
```
select name, total_views from wiki_pop sort by total_views desc limit 100;
```

TOP POPULAR ARTICLES IN LAST DAY
```
select name, last_day from wiki_pop sort by last_day desc limit 100;
```

TOP POPULAR ARTICLES IN LAST WEEK
select name, last_week from wiki_pop sort by last_week desc limit 100;

TOP POPULAR ARTICLES IN FIRST WEEK
select name, first_week from wiki_pop sort by fahoirst_week desc limit 100;

MOST POPULAR ARTICLES WITH SEARCH TERM

select name, total_views from wiki_pop where name like
"Barack%" sort by total_views desc limit 100;

TOTAL NUMBER OF HITS IN THE TIME PERIOD
select sum(total_views) from wiki_pop;

TOTAL NUMBER OF HITS IN FIRST WEEK
select sum(first_week) from wiki_pop;

TOTAL NUMBER OF HITS IN LAST WEEK
select sum(last_week) from wiki_pop;

TOTAL NUMBER OF HITS IN LAST DAY
select sum(last_day) from wiki_pop;

INCREASING TREND
select name,trend from wiki_pop sort by trend desc limit 100;

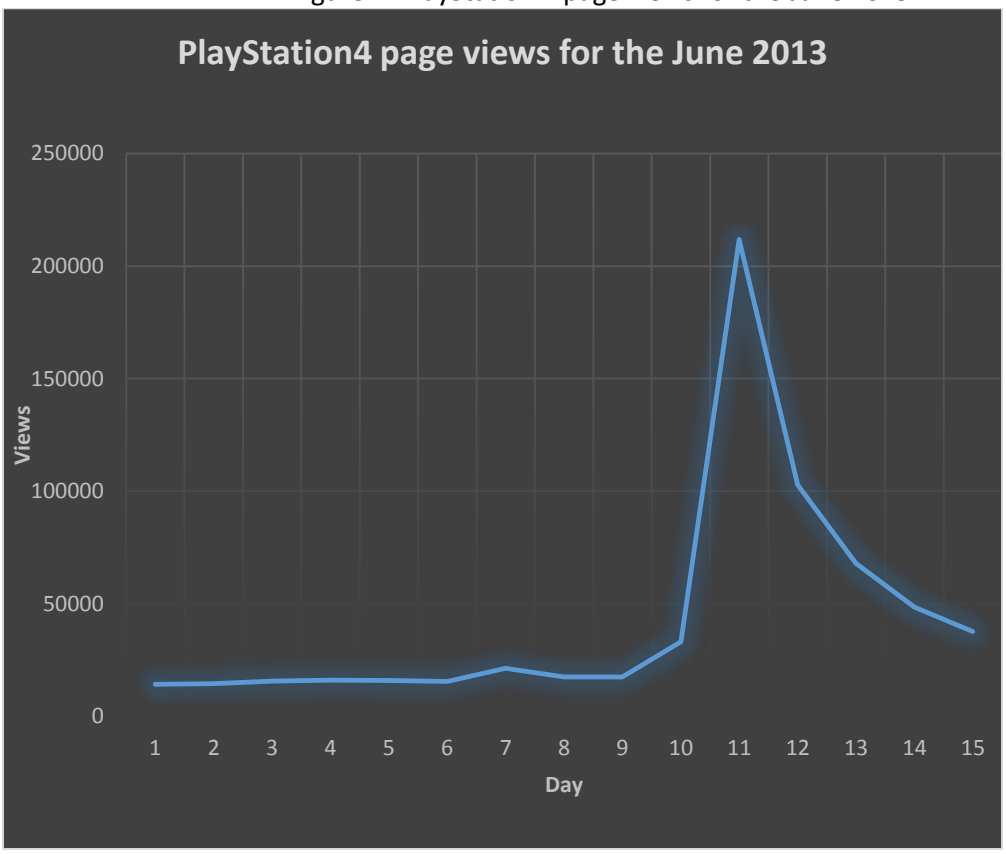
DECREASING TREND
select name,trend from wiki_pop sort by trend limit 100;

LEAST POPULAR ARTICLES
select name, total_views from wiki_pop sort by total_views limit 100;

RESULTS:

Total number of hits in the time period for all wikipedia articles = 3,182,514,682

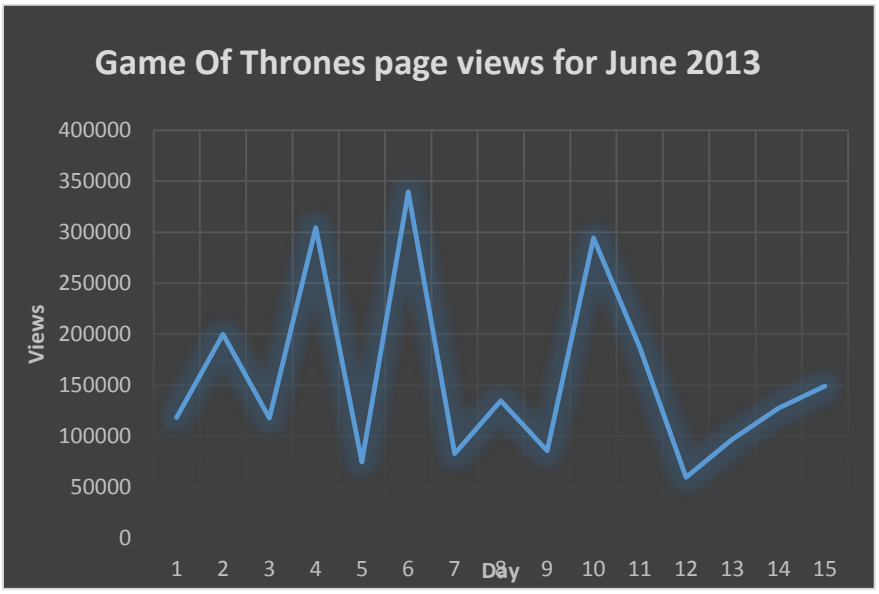
Figure 1. PlayStation 4 page views for the June 2013



Day	VIEWS
1	14363
2	14663
3	15756
4	16176
5	15947
6	15629
7	21432
8	17576
9	17580
10	33248
11	211764
12	102762
13	68046
14	48509
15	37807

From figure 1, there is rise in huge increase in the page view counts of play station 4 on 10th, 11th and 12th of June 2013, which may be because of some announcements about the play station 4.

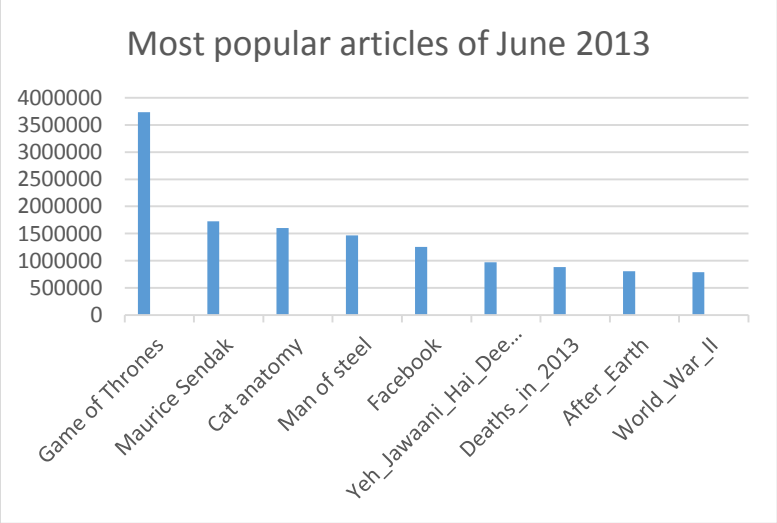
Figure 2. Game of Thrones page views for June 2013



Day	Views
1	118479
2	200025
3	117875
4	304439
5	74703
6	339496
7	82862
8	134669
9	85797
10	294452
11	187331
12	59791
13	96959
14	127459
15	149023

Figure 2, There is regular peaks and downs observed in the page views of game of thrones which indicates its overall popularity

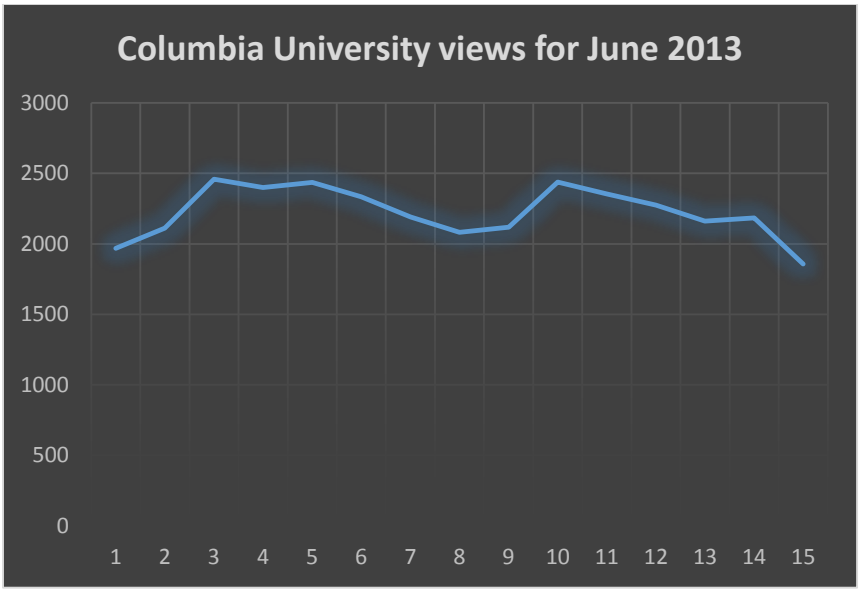
Figure 3. Most popular articles of first 15 days of June 2013.



Article	Total Views
Game of Thrones	3734604
Maurice Sendak	1722623
Cat anatomy	1603422
Man of steel	1464943
Facebook	1255734
Yeh_Jawaani_Hai_Deewani	972035
Deaths_in_2013	883352
After_Earth	805461
World_War_II	787702

Figure 3, Game of thrones has the highest page views followed by Maurice Sendak and others. This gives us a general idea of the overall popularity of various articles among people.

Figure 4: Columbia university wiki page views for June 2013



Day	Views
1	1969
2	2112
3	2459
4	2399
5	2435
6	2333
7	2191
8	2081
9	2117
10	2438
11	2355
12	2275
13	2162
14	2183
15	1858

Figure 4, there is a slight decrease in the page views of Columbia Wiki page during weekends.

CONCLUSION:

In this project, essential, need of the hour 'Big data' analysis was performed. Many interesting patterns and behaviors about Wikipedia articles were found from wiki traffic information. This kind of analysis is useful in many ways, to find out the pattern and specific behavior in particular websites or in whole of web world for particular period of time. This even has its application in log analysis, medical records analysis, science projects all of them basically have very large datasets from which essential information has to be retrieved.

FUTURE WORK:

Currently this project is not instant search. Executing hive query in turn spawns a map reduce program to bring the results. Hence it takes few min to get results for every query. This can be made to instant search by loading them in memory databases like Hbase, Amazon dynamo db. In such cases a map reduce job need not be spawned and hence results can be obtained instantly.