Wikipedia Trending Pages

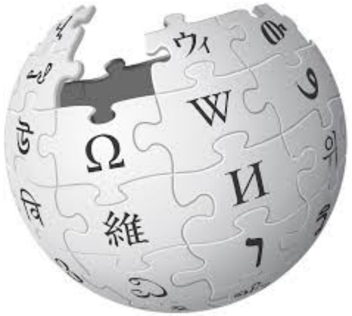
Final Project Presentation

DATASCI W251: Scaling Up! Really Big Data

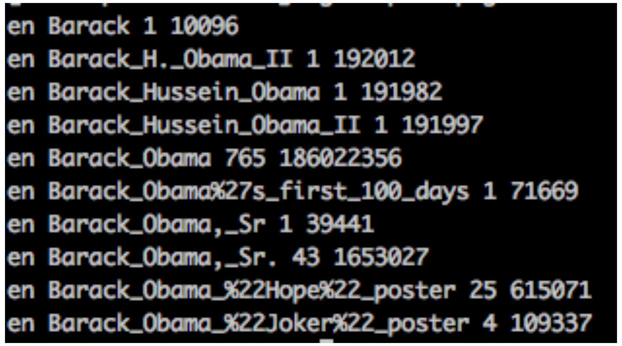
Amir Zai | Rajesh Thallam | Shelly Stanley

19th August, 2015

Project Objective

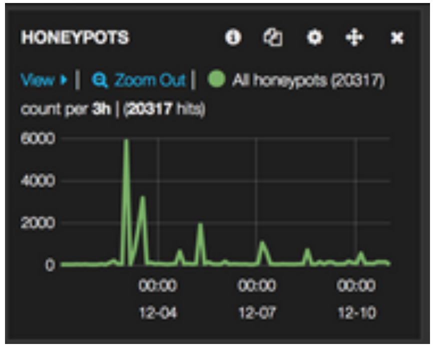


Wiki Pages



```
en Barack 1 10096
en Barack_H._Obama_II 1 192012
en Barack_Hussein_Obama 1 191982
en Barack_Hussein_Obama_II 1 191997
en Barack_Obama 765 186022356
en Barack_Obama%27s_first_100_days 1 71669
en Barack_Obama,_Sr 1 39441
en Barack_Obama,_Sr. 43 1653027
en Barack_Obama_%22Hope%22_poster 25 615071
en Barack_Obama_%22Joker%22_poster 4 109337
```

Raw Page view counts


HONEYPOTS
View ▸ | Zoom Out | ● All honeypots (20317)
count per 3h | (20317 hits)
6000
4000
2000
0
00:00
12-04
00:00
12-07
00:00
12-10

Trending Pages and Predictions

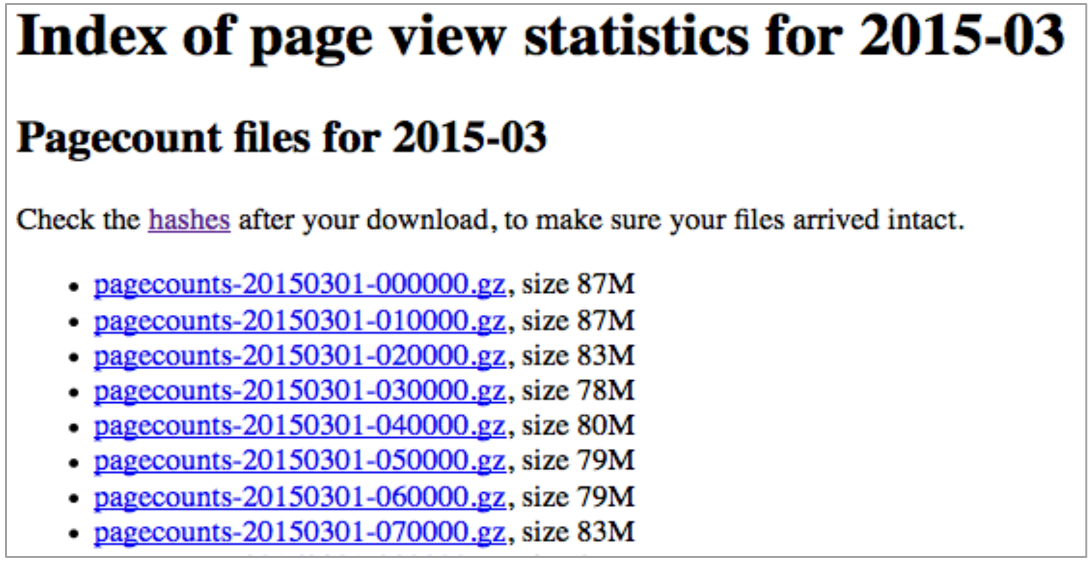
Final Goal – Wiki Trending Articles

- ✧ Top N trending pages in last 30 days
- ✧ Top N currently trending pages in last 24 hours
- ✧ Search trends for a page since Jan'15
- ✧ Predict traffic for last top N currently trending pages

Data Characteristics

Data Source – Wiki Page View Statistics

- ✧ Hourly/page aggregates of wiki page views
- ✧ Data available from 2007 till date
- ✧ For project we selected 2015 data only
  - ✧ 650 GB, compressed
  - ✧ 7+ months, Jan'15-Aug'15
  - ✧ 2.5M articles



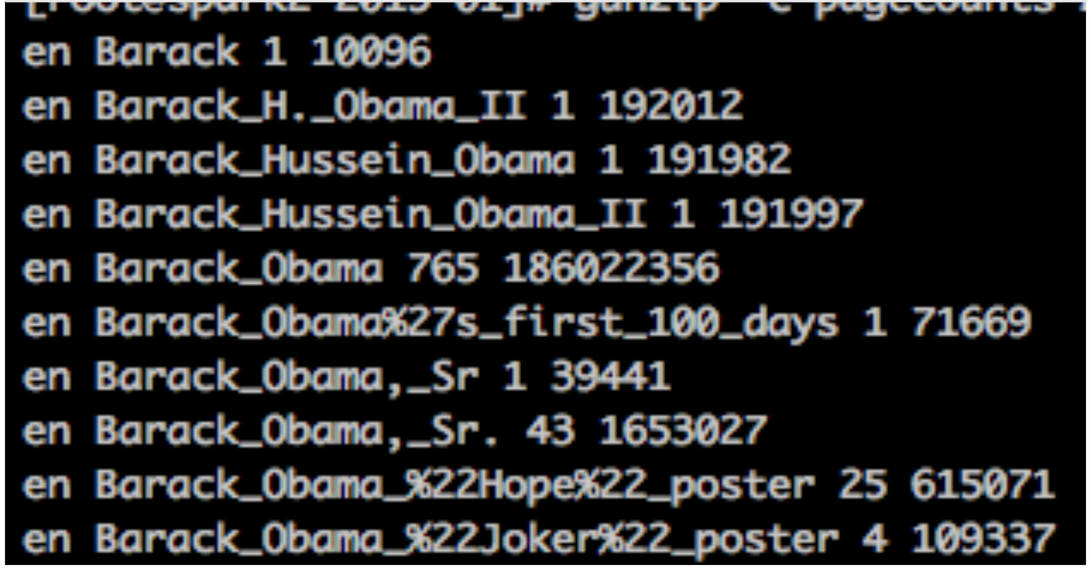
Index of page view statistics for 2015-03

Pagecount files for 2015-03

Check the hashes after your download, to make sure your files arrived intact.

- pagecounts-20150301-000000.gz, size 87M
- pagecounts-20150301-010000.gz, size 87M
- pagecounts-20150301-020000.gz, size 83M
- pagecounts-20150301-030000.gz, size 78M
- pagecounts-20150301-040000.gz, size 80M
- pagecounts-20150301-050000.gz, size 79M
- pagecounts-20150301-060000.gz, size 79M
- pagecounts-20150301-070000.gz, size 83M

Data Source – Wiki Page View Statistics

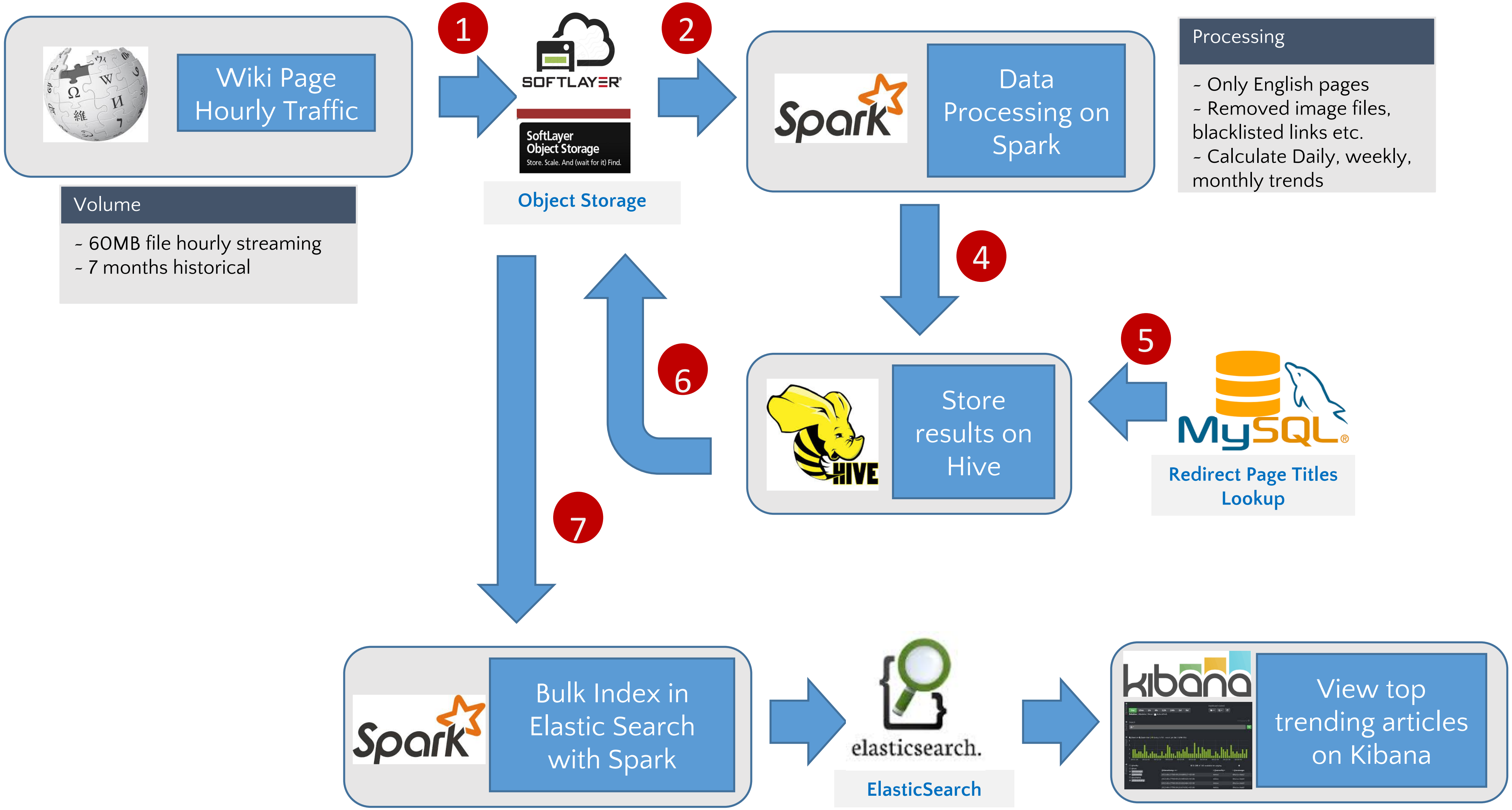


```
en Barack 1 10096
en Barack_H._Obama_II 1 192012
en Barack_Hussein_Obama 1 191982
en Barack_Hussein_Obama_II 1 191997
en Barack_Obama 765 186022356
en Barack_Obama%27s_first_100_days 1 71669
en Barack_Obama,_Sr 1 39441
en Barack_Obama,_Sr. 43 1653027
en Barack_Obama_%22Hope%22_poster 25 615071
en Barack_Obama_%22Joker%22_poster 4 109337
```

Data fields of interest

- ✧ Date and hour on the file name
- ✧ Fields in the file
  - ✧ Wiki project name
  - ✧ Page title
  - ✧ Page views in a particular hour

System Architecture



Wiki Page Hourly Traffic

Volume

- 60MB file hourly streaming
- 7 months historical

1

Object Storage

2

Data Processing on Spark

Processing

- Only English pages
- Removed image files, blacklisted links etc.
- Calculate Daily, weekly, monthly trends

4

Store results on Hive

5

Redirect Page Titles Lookup

6

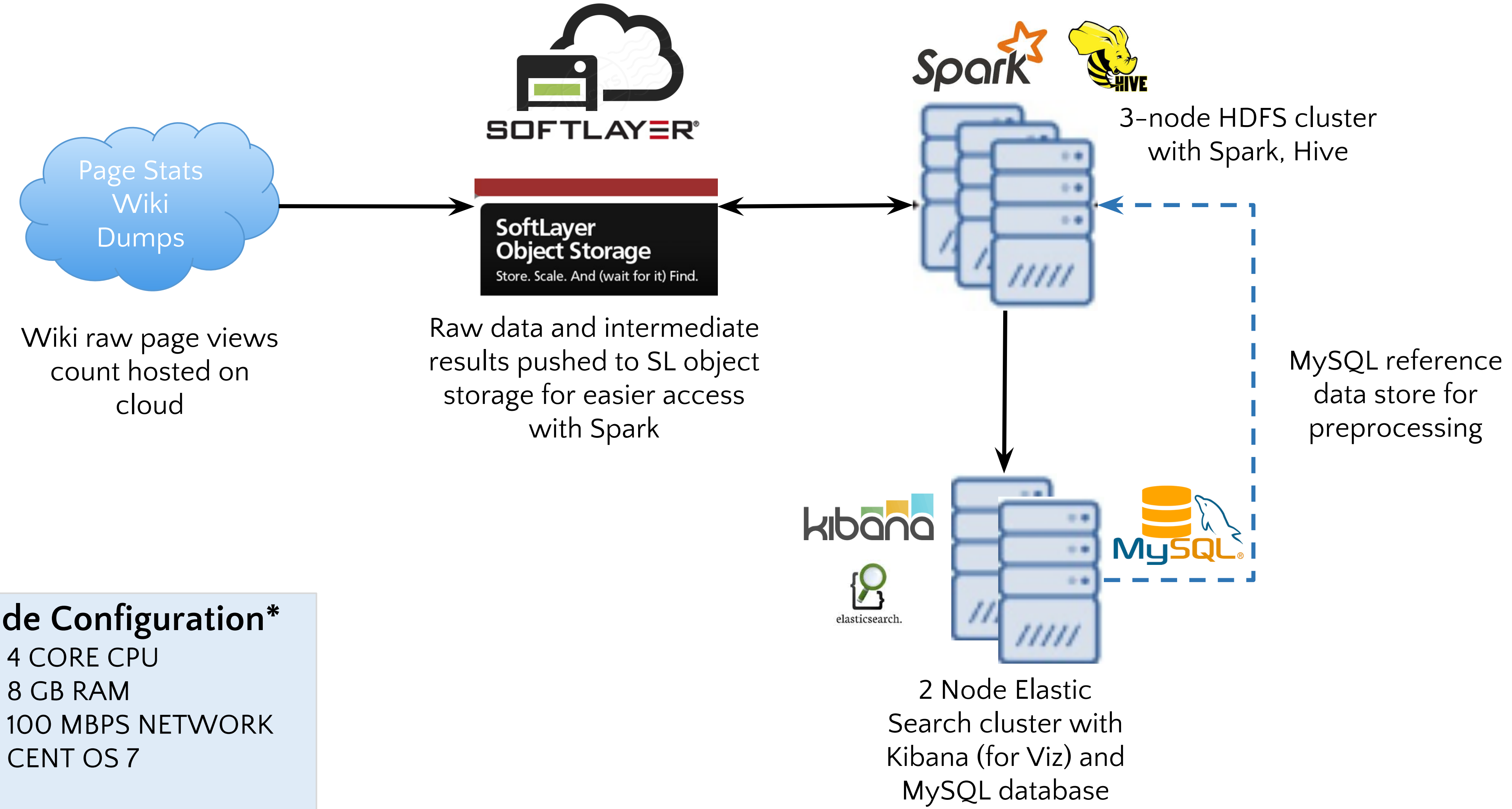
7

Bulk Index in Elastic Search with Spark

ElasticSearch

View top trending articles on Kibana

Cluster and Systems Infrastructure



Wiki raw page views count hosted on cloud

Raw data and intermediate results pushed to SL object storage for easier access with Spark

3-node HDFS cluster with Spark, Hive

MySQL reference data store for preprocessing

2 Node Elastic Search cluster with Kibana (for Viz) and MySQL database

Node Configuration*

4 CORE CPU
8 GB RAM
100 MBPS NETWORK
CENT OS 7

* All node are in SJC01 data center on the same VLAN

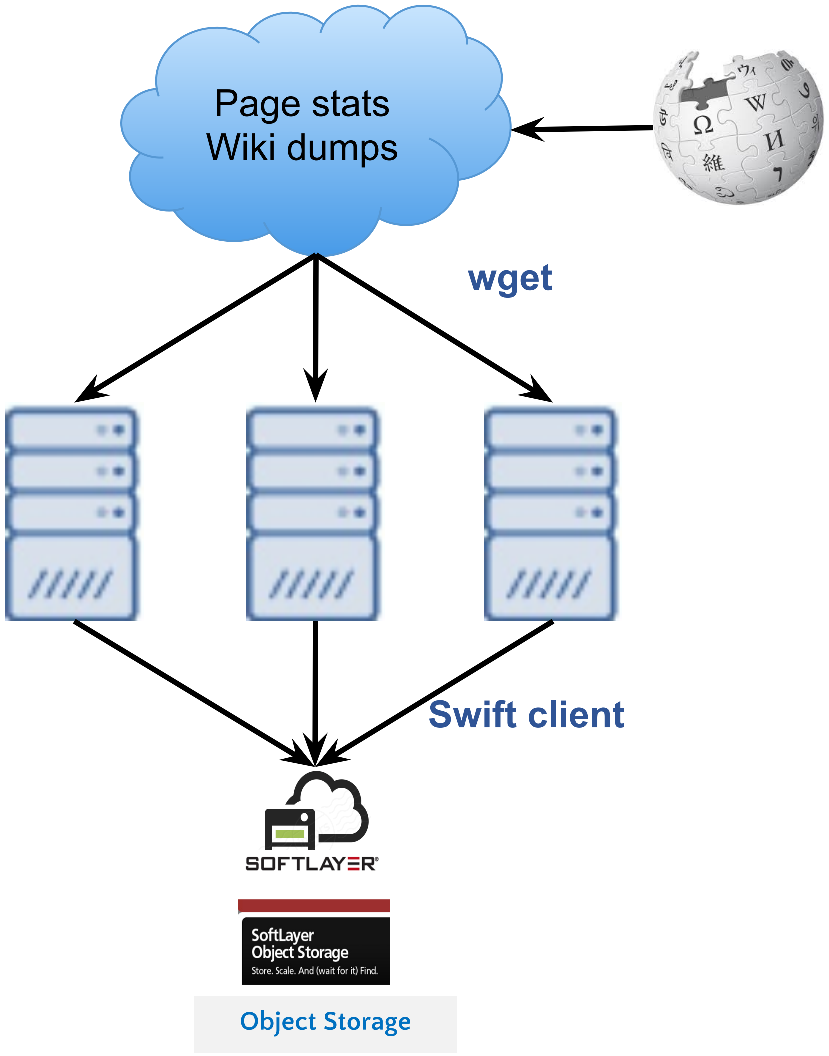
Data Ingestion

Data Ingestion

- Upload all data files to SoftLayer Object Storage
- About 60GB of data per month
- One file with counts for each hour
- Used 3-node cluster to speed this up
- Shell script to wget files and upload to object storage using python-swiftclient

Why Object Storage?

- Not our original choice
- Max open files issue on HDFS even after changing limits
- Got the issue away with Swift



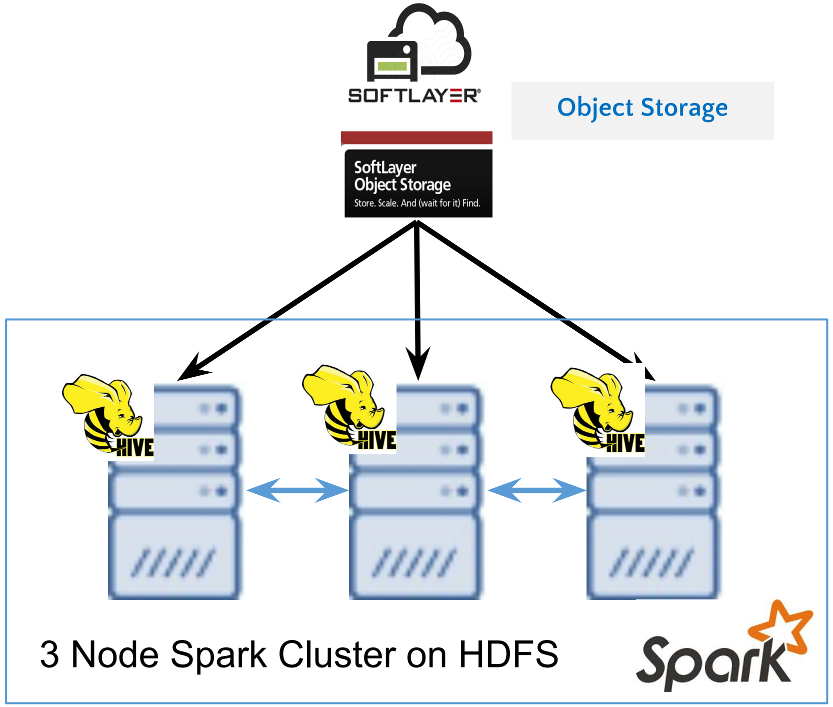
Data Cleansing & Processing

Data Cleansing

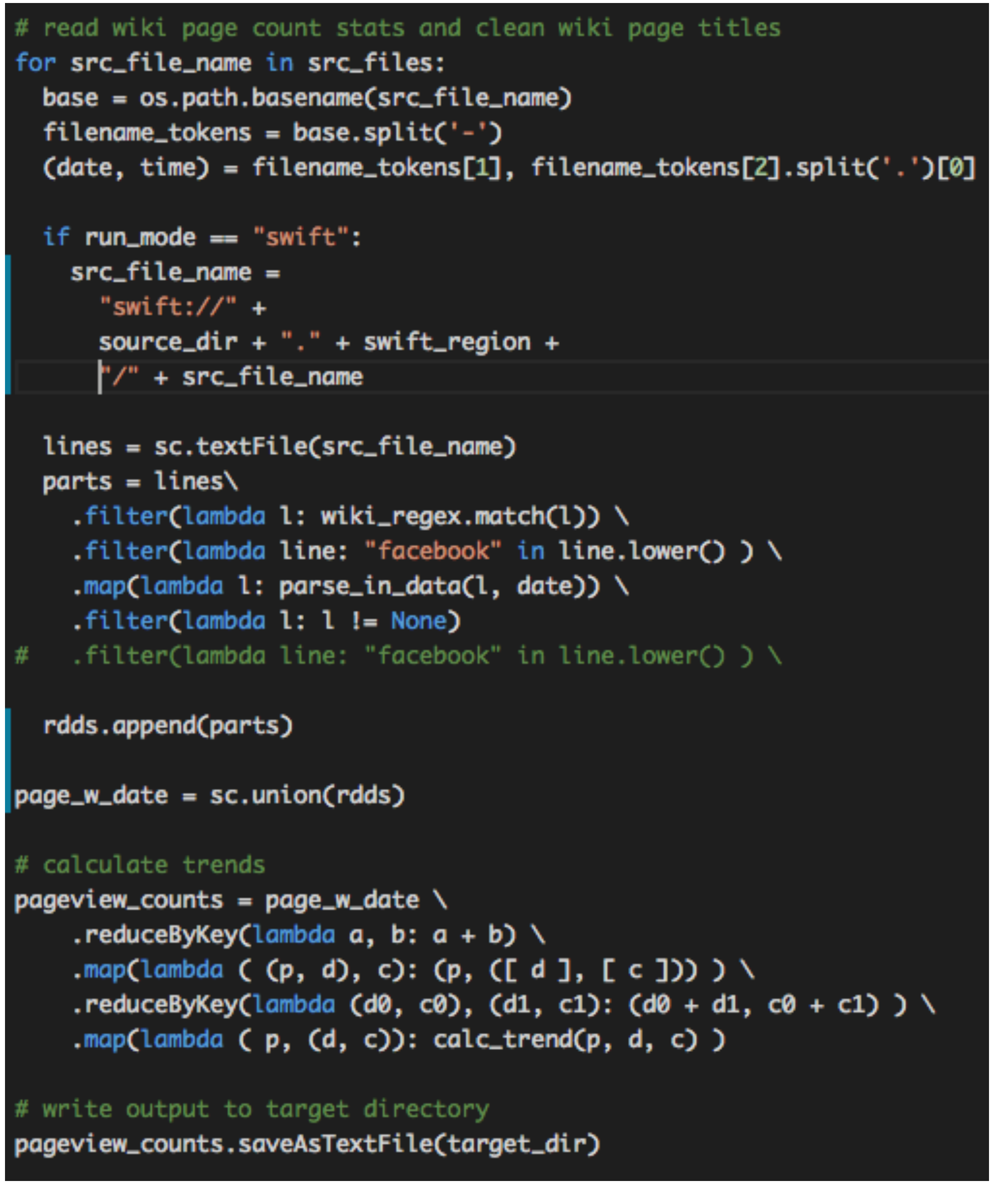
- ✧ Read each file on the object storage
- ✧ Only kept English (en) pages in each file
- ✧ Removed hits on image files, noisy links etc.
- ✧ Cleaned redirects to the same page

Data Processing

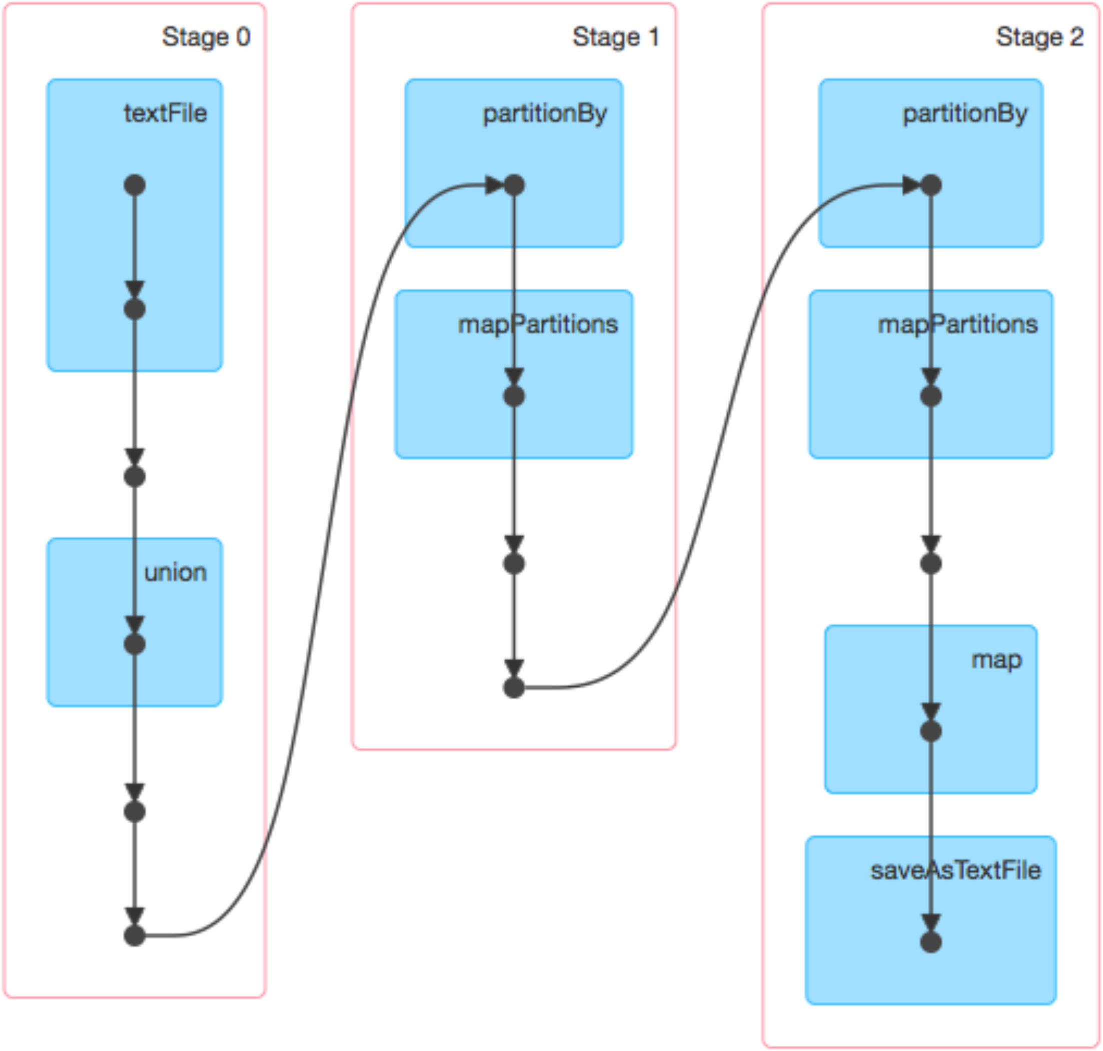
- ✧ Calculated daily, weekly and monthly trends
- ✧ Store the intermediate results on HDFS
- ✧ Load results into Hive data store for further processing


SOFTLAYER
Object Storage
SoftLayer Object Storage
Store. Scale. And (wait for it) Find.
HIVE
HIVE
HIVE
3 Node Spark Cluster on HDFS
Spark

Apache Spark DAG Visualization

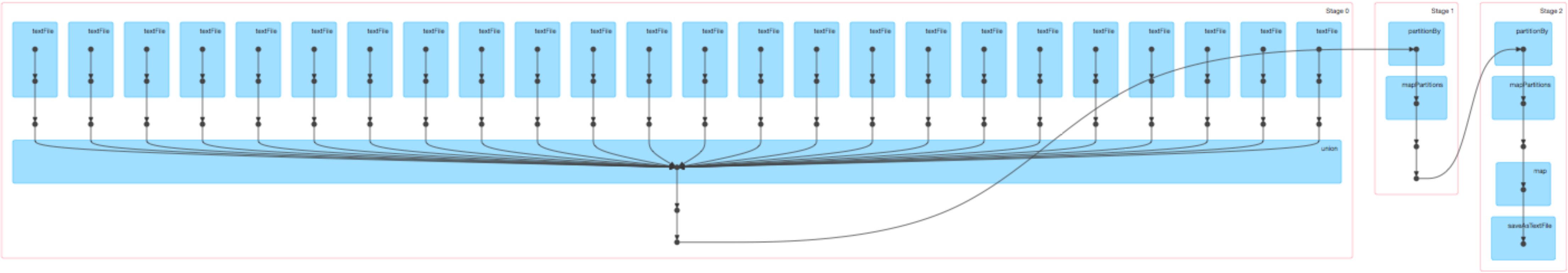


```
# read wiki page count stats and clean wiki page titles
for src_file_name in src_files:
  base = os.path.basename(src_file_name)
  filename_tokens = base.split('-')
  (date, time) = filename_tokens[1], filename_tokens[2].split('.')[0]

  if run_mode == "swift":
    src_file_name =
      "swift://" +
      source_dir + "." + swift_region +
      "/" + src_file_name

  lines = sc.textFile(src_file_name)
  parts = lines\
    .filter(lambda l: wiki_regex.match(l)) \
    .filter(lambda line: "facebook" in line.lower() ) \
    .map(lambda l: parse_in_data(l, date)) \
    .filter(lambda l: l != None)
#   .filter(lambda line: "facebook" in line.lower() ) \

  rdds.append(parts)

page_w_date = sc.union(rdds)

# calculate trends
pageview_counts = page_w_date \
    .reduceByKey(lambda a, b: a + b) \
    .map(lambda ( (p, d), c): (p, ([ d ], [ c ])) ) \
    .reduceByKey(lambda (d0, c0), (d1, c1): (d0 + d1, c0 + c1) ) \
    .map(lambda ( p, (d, c)): calc_trend(p, d, c) )

# write output to target directory
pageview_counts.saveAsTextFile(target_dir)
```



Apache Spark DAG Visualization

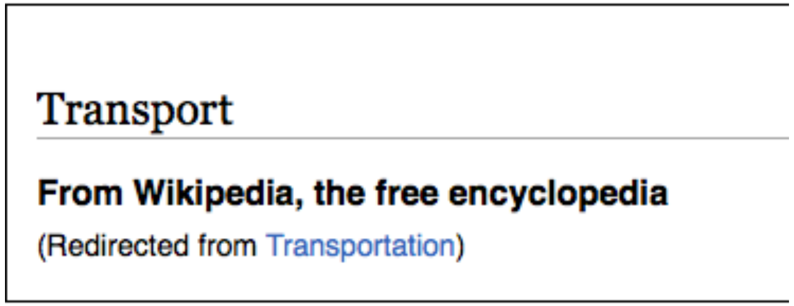
DAG Visualization for a batch of 1 day i.e. 24 files (1 file per hour in a day)



Merging Historical with Current Trends

Data Processing

- Before merging further data cleansing required on page titles

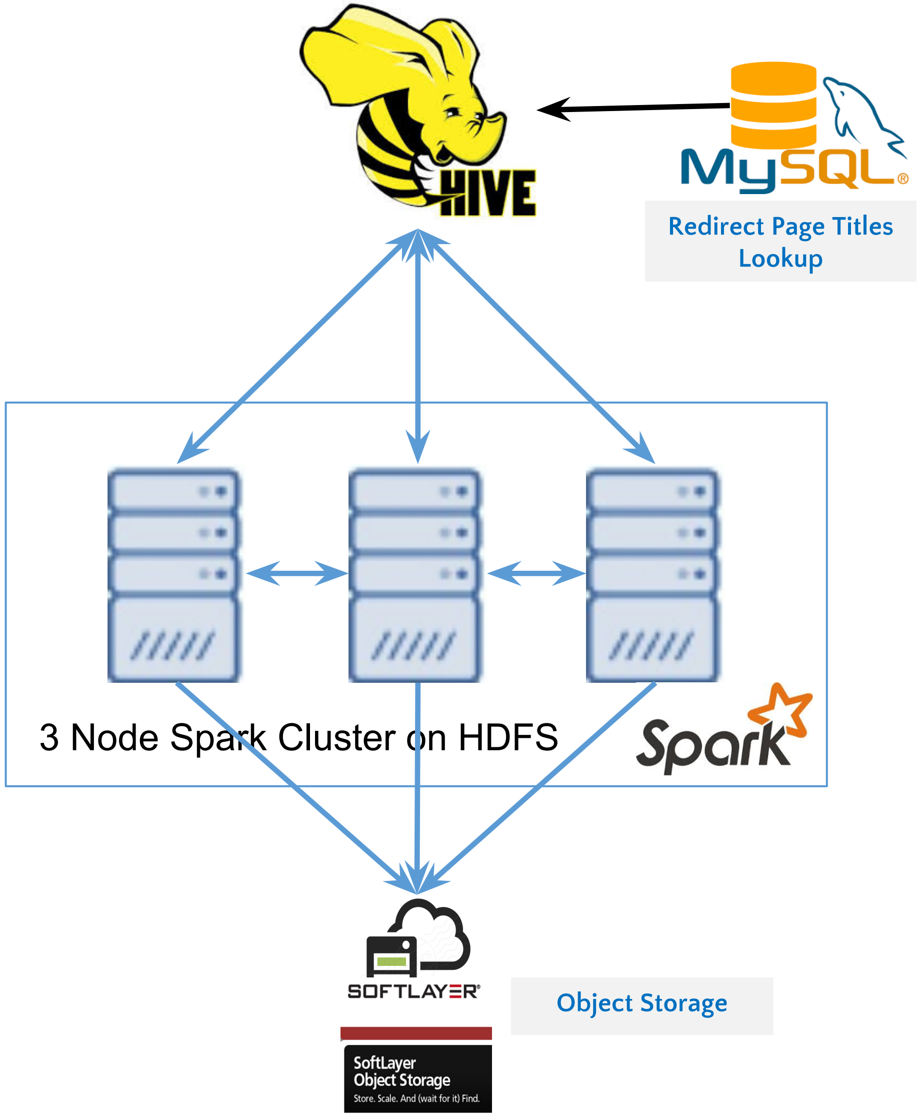


Transport

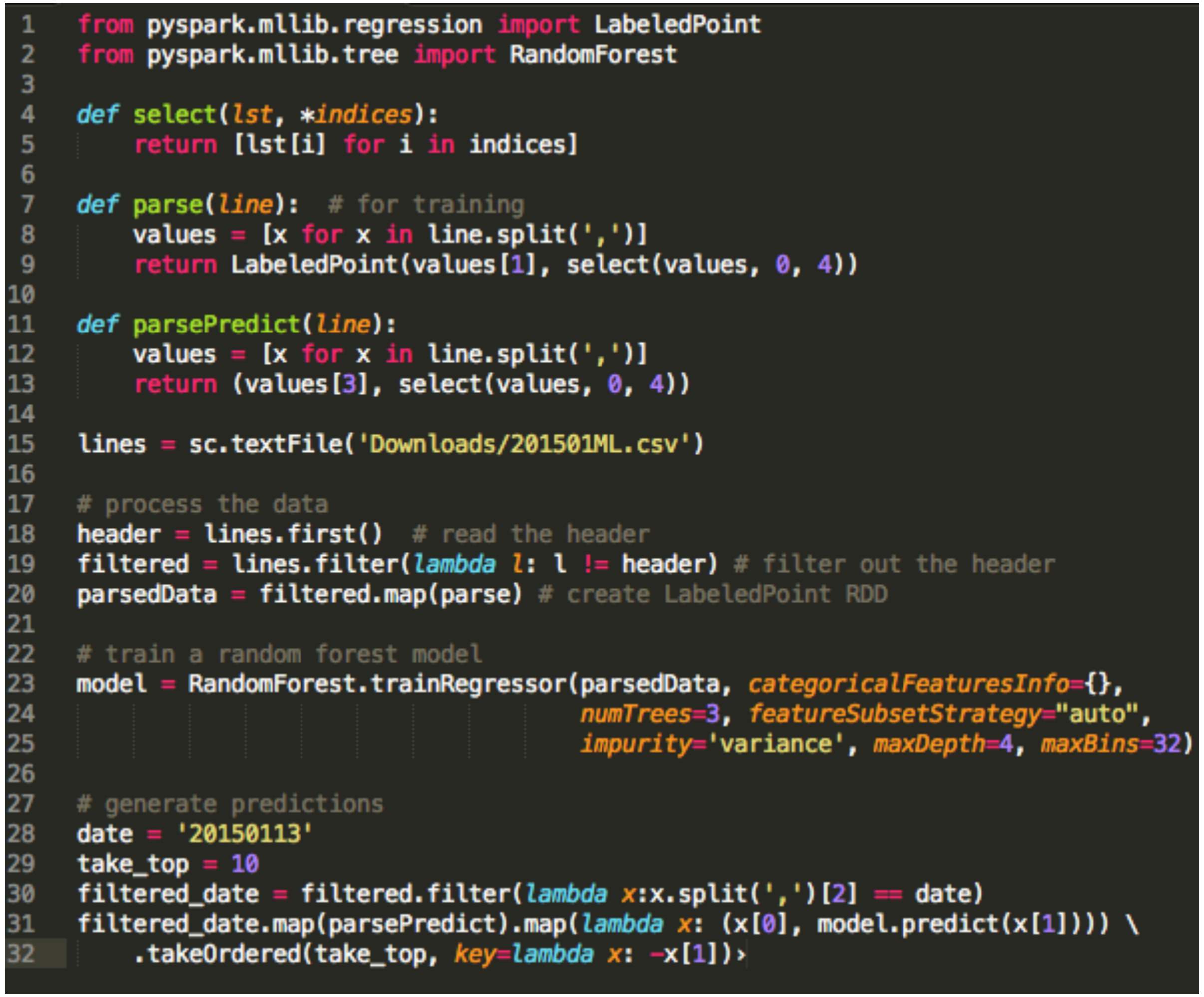
From Wikipedia, the free encyclopedia

(Redirected from Transportation)

- MySQL scripts from wiki dumps to correct for redirects
- Run Spark to merge latest results with historical results
- The final merged results on HDFS is fed to Swift Object Storage and Hive data store


HIVE
MySQL
Redirect Page Titles Lookup
3 Node Spark Cluster on HDFS
Spark
SOFTLAYER
Object Storage
SoftLayer Object Storage
Store. Scale. And (wait for it) Find.

Predicting Traffic for Top N Trending Pages



```python
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.tree import RandomForest

def select(lst, *indices):
    return [lst[i] for i in indices]

def parse(line):  # for training
    values = [x for x in line.split(',')]
    return LabeledPoint(values[1], select(values, 0, 4))

def parsePredict(line):
    values = [x for x in line.split(',')]
    return (values[3], select(values, 0, 4))

lines = sc.textFile('Downloads/201501ML.csv')

# process the data
header = lines.first()  # read the header
filtered = lines.filter(lambda l: l != header) # filter out the header
parsedData = filtered.map(parse) # create LabeledPoint RDD

# train a random forest model
model = RandomForest.trainRegressor(parsedData, categoricalFeaturesInfo={},
                                    numTrees=3, featureSubsetStrategy="auto",
                                    impurity='variance', maxDepth=4, maxBins=32)

# generate predictions
date = '20150113'
take_top = 10
filtered_date = filtered.filter(lambda x:x.split(',')[2] == date)
filtered_date.map(parsePredict).map(lambda x: (x[0], model.predict(x[1]))) \
    .takeOrdered(take_top, key=lambda x: -x[1])
```

- Random Forest Regression (R2 = 0.67)
- Predict traffic percentage for the next day based on daily and weekly trend features
- Sort and take top N

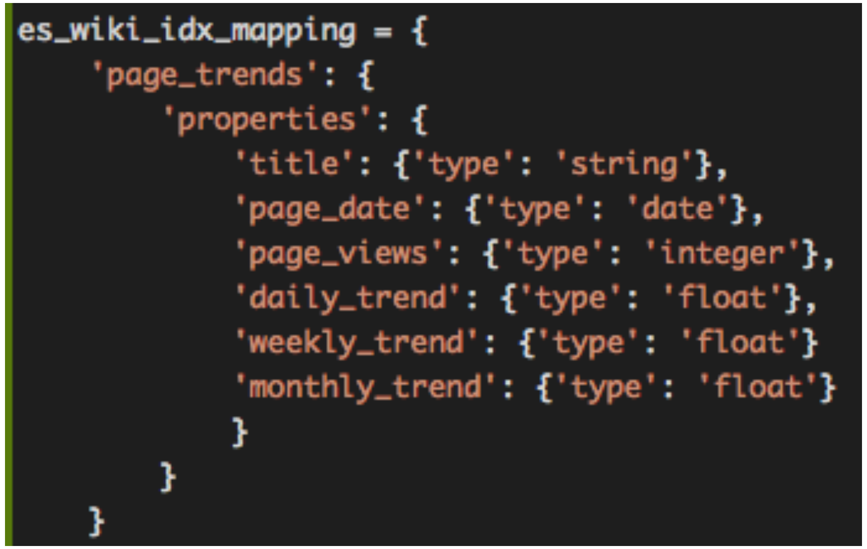
Example: Top 10 for January 14, 2015

Page	Pred
American_Sniper_(film)	4.292847
Transparent_(TV_series)	3.853456
Cristiano_Ronaldo	2.921265
Edward_Norton	1.842253
Genghis_Khan	1.719706
Lucy_(2014_film)	1.300001
Snowpiercer	1.281999
Penny_Dreadful_(TV_series)	1.174609
Michael_Keaton	1.153589
Ernest_Hemingway	1.129073

Search with ElasticSearch

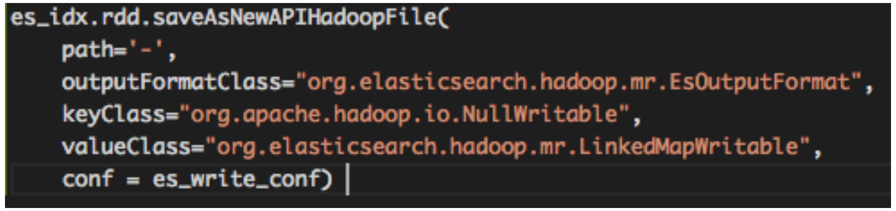
Building Search Index with Elastic Search

- ✧ Final result format

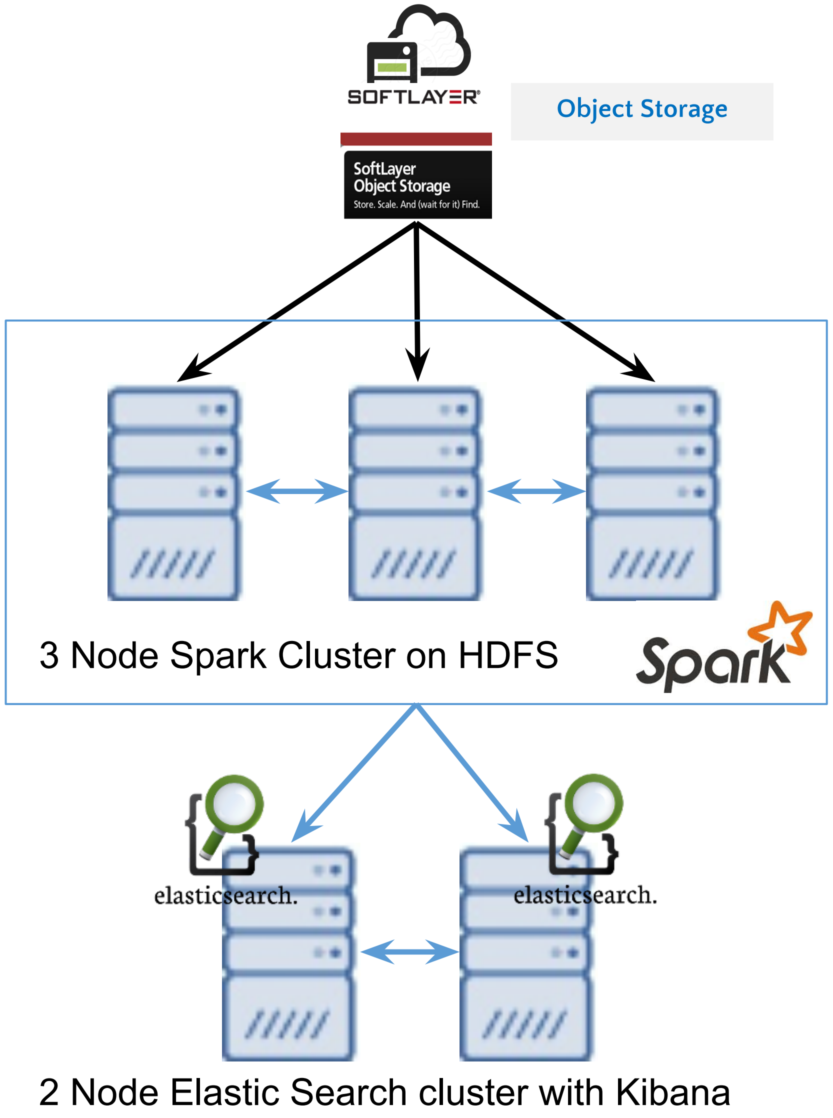


```
es_wiki_idx_mapping = {
    'page_trends': {
        'properties': {
            'title': {'type': 'string'},
            'page_date': {'type': 'date'},
            'page_views': {'type': 'integer'},
            'daily_trend': {'type': 'float'},
            'weekly_trend': {'type': 'float'}
            'monthly_trend': {'type': 'float'}
        }
    }
}
```

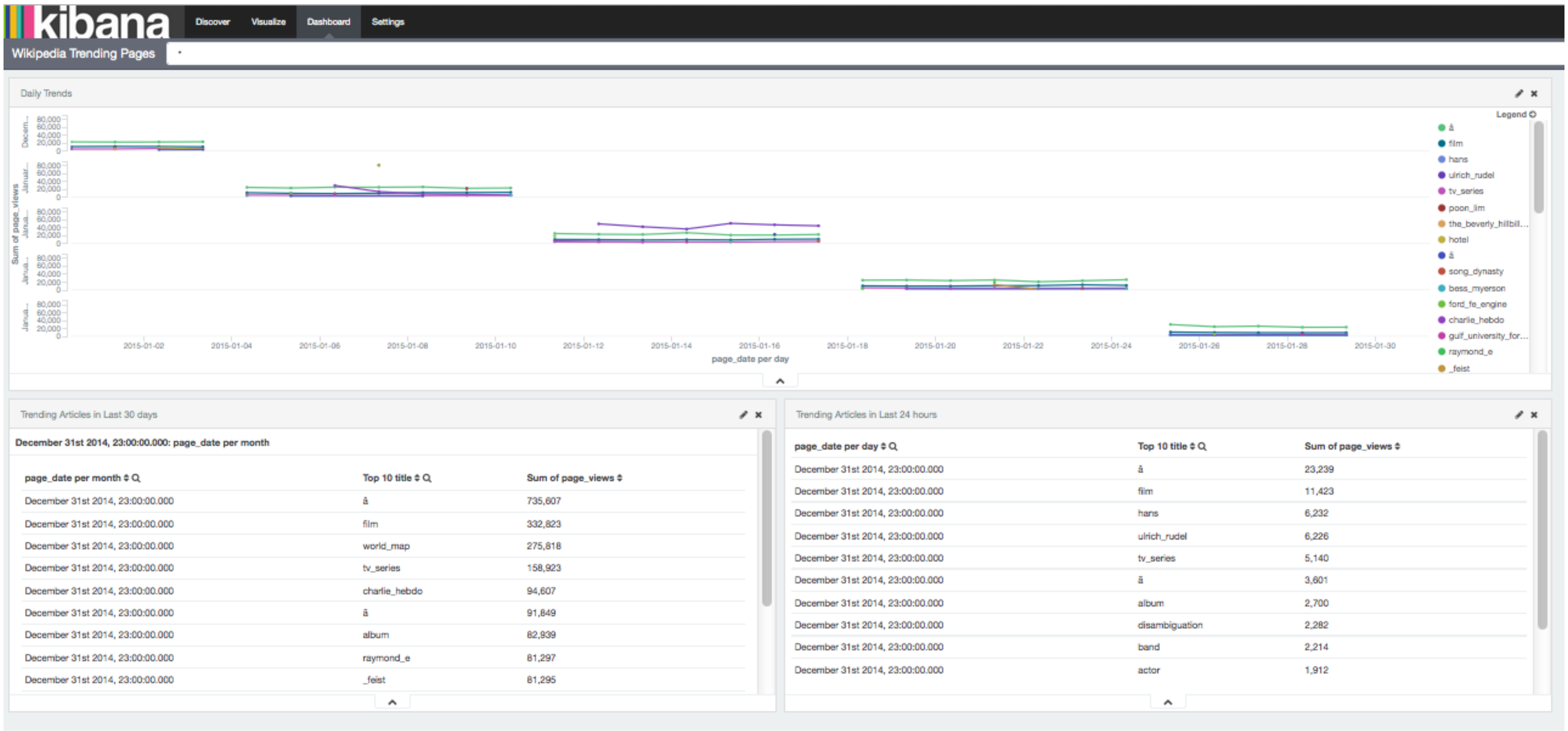
- ✧ Bulk indexing with Spark



```
es_idx.rdd.saveAsNewAPIHadoopFile(
    path='-',
    outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
    keyClass="org.apache.hadoop.io.NullWritable",
    valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",
    conf = es_write_conf)
```


SOFTLAYER
Object Storage
SoftLayer Object Storage
Store. Scale. And (wait for it) Find.
3 Node Spark Cluster on HDFS
Spark
elasticsearch.
elasticsearch.
2 Node Elastic Search cluster with Kibana

Trend Analysis with Kibana



Trending Articles in Last 30 days

December 31st 2014, 23:00:00.000: page_date per month

page_date per month ⇕ 🔍	Top 10 title ⇕ 🔍	Sum of page_views ⇕
December 31st 2014, 23:00:00.000	ā	735,607
December 31st 2014, 23:00:00.000	film	332,823
December 31st 2014, 23:00:00.000	world_map	275,818
December 31st 2014, 23:00:00.000	tv_series	158,923
December 31st 2014, 23:00:00.000	charlie_hebdo	94,607
December 31st 2014, 23:00:00.000	ā	91,849
December 31st 2014, 23:00:00.000	album	82,939
December 31st 2014, 23:00:00.000	raymond_e	81,297
December 31st 2014, 23:00:00.000	_feist	81,295

Trending Articles in Last 24 hours

page_date per day ⇕ 🔍	Top 10 title ⇕ 🔍	Sum of page_views ⇕
December 31st 2014, 23:00:00.000	ā	23,239
December 31st 2014, 23:00:00.000	film	11,423
December 31st 2014, 23:00:00.000	hans	6,232
December 31st 2014, 23:00:00.000	ulrich_rudel	6,226
December 31st 2014, 23:00:00.000	tv_series	5,140
December 31st 2014, 23:00:00.000	ā	3,601
December 31st 2014, 23:00:00.000	album	2,700
December 31st 2014, 23:00:00.000	disambiguation	2,282
December 31st 2014, 23:00:00.000	band	2,214
December 31st 2014, 23:00:00.000	actor	1,912

Wikipedia Trending Pages

Challenges

1

Too much noise in source file

- Date is available on the file name and the rest in the file so all the files cannot be ingested at same time
- Cleaning the page titles – too much noise

2

Err – Max files open issue

- HDFS throws error max files open issue even after setting max open files high
- Stored everything on SL Objet Storage

3

Spark, Hadoop and Swift Integration

- By default Hadoop and Swift integration expects Keystone authentication
- To make it work with SL Object Storage a patch has to be applied on Hadoop

4

Not so friendly Kibana

- Kibana 1.4 – Not much customization possible on x and y axis in trend analysis
- Realized late – ElasticSearch does not detect date data type automatically

Future Improvements

Additional Sources for better prediction

Spark SQL Data Frame

Instead of Hive

Country wise Trend Analysis

currently only en pages

Dynamic Cluster Balancing

Instead of fixed size cluster

Better User Interface

More features for a more accurate prediction model

Thank You!