Top-K words on Hadoop Cluster

Objective:-

The objective of this assignment is to find top-K words from large dataset using MapReduce and Hive. We also analyzed the performance by changing default configuration of Hadoop file system.

Problem Statement:-

- Determine 100 most frequent words in the given dataset using MapReduce and Hive.
- Determine 100 most frequent words having more than 3 characters in the given dataset using MapReduce and Hive.

Input Dataset:-

1. data_32GB.txt – A text file of size 32GB
2. data_64GB.txt - A text file of size 64GB

1.MapReduce

a) We started with the most basic mapreduce code.
   i) Read a file and stream data from mapper.
   ii) Reducer will collect streamed data and combine it and store the data in dictionary.
   iii) User heapq to find top-K words from dictionary. Each reducer will have top-K words
   iv) Combine these outputs and find top-K words.

b) We used combiner to lower workload on reducer.
   i) Read a file, combine the frequency of same words and stream data from mapper.
   ii) Reducer will collect streamed data and combine it and store the data in dictionary.

iii) User heapq to find top-K words from dictionary. Each reducer will have top-K words

iv) Combine these outputs and find top-K words.

Code and Output:

Approach 1: (Without Combiner)

mapper1.py × mapper.py reduce.py reducer1.py

```python
import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print('%s\t%s' % (word, 1))
```

mapper.py

mapper1.py mapper.py reduce.py reducer1.py ×

```python
import operator
import sys
import heapq

current_word = None
current_count = 0
word = None
reducer_dict = {}
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    if(line=''):
        break
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    reducer_dict[word] = reducer_dict.get(word,0) + 1

k_keys_sorted = heapq.nlargest(100, reducer_dict,key=reducer_dict.get)
for k in k_keys_sorted:
    print(k,reducer_dict[k])
```

reducer.py

Output:-

```
the      142487086
quot     131585830
and      69220506
title    61488231
ref      51751180
amp      49225842
text     49075004
page     43551751
User     39032500
http     31158548
The      30290149
format   30216932
timestamp        29739717
model    29146823
revision         28976784
contributor      28877146
sha      28819730
username         28732925
for      28595464
Category         28259946
comment  28177273
talk     27478425
com      26699461
www      25977261
name     25490183
was      22417581
org      21679663
parentid         20858357
         20676353
date     18870014
that     17894127
with     17732012
style    17065159
space    16651419
from     16130333
wiki     16005668
align    15562284
web      14532426
xml      14515617
preserve         14445312
wikitext         14410739
small    14201593
url      13890427
UTC      13541041
cite     12501452
Wikipedia        12328698
php      11879395
span     11633046
```

```
has      6824392
user     6524165
news     6426159
minor    6406500
New      6332496
United   6166300
have     6161295
nbsp     6146603
which    6036545
you      5956112
left     5954204
domain   5923750
deletion         5912897
type     5859906
team     5827815
but      5787673
also     5590499
archive  5512341
discussion       5426820
year     5413914
flagicon         5339690
Special  5202424
time     5195381
were     5120157
sup      5077419
other    5028623
one      4964632
class    4929818
WUL      4891112
```

(a).32GB file output

```
the      281548636
quot     271085749
and      136838428
title    123735761
ref      111849091
text     99382349
page     86694594
amp      83259619
User     76064278
format   61105910
The      59786522
timestamp        58997425
model    58700895
revision         58424690
username         58212679
contributor      58182320
sha      58073343
Category         57342274
for      57340175
comment  56598539
http     56333267
com      55337415
name     53819828
talk     53751334
www      50376680
was      45039361
         44645254
date     41252680
parentid         39373877
style    37672589
```

```
style    37672589
org      36110201
that     35303009
with     34877621
align    33993131
from     32951836
space    32583942
wiki     32385619
url      31878758
web      30764208
xml      29224889
preserve         29103817
wikitext         29048235
cite     28550304
small    26620798
UTC      25680193
span     24669608
Wikipedia        24318818
not      22113758
this     21739160
center   21576645
first    21533201
font     21352183
accessdate       21061231
https    20784549
redirect         20483358
color    20287776
link     19089417
publisher        17904347
COIBot   17751367
are      16821126
php      16798310
wikipedia        16431343
article  15767741
his      15630919
last     15276704
background       15235769
index    15199776
team     14729521
news     14223895
diff     13962722
left     13866877
image    13663678
has      13599852
html     13414988
domain   13163006
flagicon         12592406
nbsp     12351719
you      12251849
```

(b) 64GB file output

Approach 2: (With Combiner)

- On a large dataset when we run MapReduce job, large chunks of intermediate data is generated by the mapper and streamed to the reducer for further processing. This might lead to enormous network congestion. MapReduce framework provides a function known as Hadoop Combiner that plays a key role in reducing the congestion.
- The logic of combiner is added at the end of the code which will combine the tuples based on the first parameter(word) and will stream the data to the reducer.

mapper1.py | mapper.py × | reduce.py | reducer1.py

```python
import sys
import itertools
import operator

map_list = []

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        #Create tuples with first parameter as a key and second parameter as a value (default=1)
        map_list.append((word, 1))
#combine tuples with same first parameter and stream the output to the reducer
for group in itertools.groupby(map_list, operator.itemgetter(0)):
        print(group[0],'\t',len(list(map(operator.itemgetter(1), group[1]))))
```

mapper.py (with combiner)

```python
import operator
import sys
import heapq

current_word = None
current_count = 0
word = None
reducer_dict = {}
# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t')
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue

    reducer_dict[word] = reducer_dict.get(word,0) + count

k_keys_sorted = heapq.nlargest(100, reducer_dict,key=reducer_dict.get)
for k in k_keys_sorted:
    print(k,reducer_dict[k])
```

reducer.py

Execution:-

- To execute MapReduce in python, we have to use hadoop-streaming jar file. Below is the command to execute MapReduce job in Hadoop.

```
hadoop jar /home/bigdata12/assignment2jar/hadoop-streaming-2.4.0.jar
-D mapreduce.map.memory.mb=8192
-D  mapreduce.reduce.memory.mb=8192
-file /home/bigdata12/assignment2jar/mapred/mapper.py -mapper "python mapper.py"
-file /home/bigdata12/assignment2jar/mapred/reduce.py -reducer "python reduce.py"
-input ./data_32GB.txt -output ./output
```

1) jar file(hadoop-streaming-2.4.0.jar)
2) -D configurations tuning
3) -file (file path)
4) -mapper (consider the file as mapper and execute the command python mapper.py)
5) -reducer (consider the file as reducer and execute the command python reducer.py)
6) -input (file which will to be processed to find top-K words)
7) -output (directory where all the processed reducers will be stored)

After executing above command, hadoop will generate the output based on the reducer tasks. Each task will have local top-K words. We need to process these output files and

find global top-K words. Below command can be used to find global top-K words from the file.

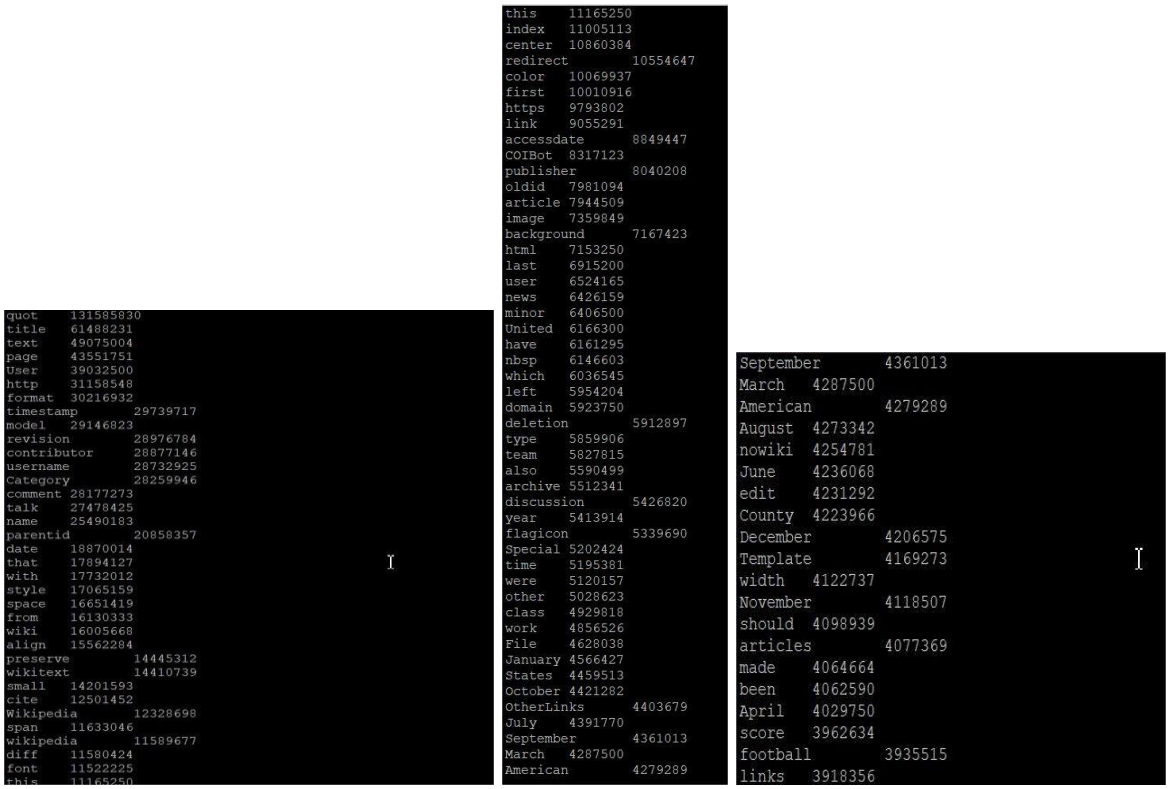
```
hadoop fs -cat /output/of/wordcount/part* | sort -n -k2 -r | head -n K
```

<u>Approach 3:- (To find words having length more than 3)</u>

To find words having length more than 3, there are two possible approaches.

1) Filter out words from mapper:- This approach will stream less data from mapper to reducer.
2) Filter out data in Reducer:- This approach will stream same data as approach 1 but will filter out data at the time of storing data into output file.

We have followed first approach and removed the words before streaming to reducer.



Output of 32GB file for words having length > 3

August	10035550
March	9776756
July	9300882
April	9259785
width	9189153
other	9158537
January	9118251
December	9045832
Special	8947655
September	8940071
Template	8789580
October	8689445
November	8603425
June	8459641
American	8430486
County	8349573
made	8328269
been	8323094
should	8219486
football	8189429
States	8109504
February	7782588
user	7742295
about	7704881
right	7593353
list	7493602

quot	271085749
title	123735761
text	99382349
page	86694594
User	76064278
format	61105910
timestamp	58997425
model	58700895
revision	58424690
username	58212679
contributor	58182320
Category	57342274
comment	56598539
http	56333267
name	53819828
talk	53751334
date	41252680
parentid	39373877
style	37672589
that	35303009
with	34877621
align	33993131
from	32951836
space	32583942
wiki	32385619
preserve	29103817
wikitext	29048235
cite	28550304

cite	28550304
small	26620798
span	24669608
Wikipedia	24318818
this	21739160
center	21576645
first	21533201
font	21352183
accessdate	21061231
https	20784549
redirect	20483358
color	20287776
link	19089417
publisher	17904347
COIBot	17751367
wikipedia	16431343
article	15767741
last	15276704
background	15235769
index	15199776
team	14729521
news	14223895
diff	13962722
left	13866877
image	13663678
html	13414988
domain	13163006
flagicon	12592406
nbsp	12351719
have	11920128
deletion	11882318
which	11810297
discussion	11526108
United	11499865
oldid	11354677
type	11019543
work	11014622
also	11005633
score	10904020
class	10894094
minor	10842971
year	10798170
time	10455893
File	10416546
OtherLinks	10326099
archive	10213975
were	10192131
August	10035550

Output of 64 GB file for words having length > 3

2.**Hive**

Approach:

a) Top-K words
   1. Temporary storing the data into a table with only one column i.e for each line of the text document. (32GB and 64GB).
   2. Now creating another table with two columns, one is the word and other is count of that word. Used explode function in HQL to break each word in the line of the temporary table.
   3. Run the select query to get the top 100 most repeating words by ordering count in descending order and limiting the result to 100.

b) Limiting to words having more than 3 characters
   1) Temporary storing the data into a table with only one column i.e for each line of the text document. (32GB and 64GB).

2) Now creating another table with two columns, one is the word and other is count of that word. Used explode function in HQL to break each word in the line of the temporary table.
3) Run the select query to get the top 100 most repeating words by ordering count in descending order and limiting the result to 100 and also limiting words having more than 3 characters.

a) Top-K Words:

i) Output of 32GB file

```
hive> select * from final_word_count_32_first_display
    > order by count desc
    > limit 100;
Query ID = bigdata12_20190512222222_a53a3171-5304-4e45-9a07-8f596606887e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
```

```
Ended Job = job_1557022631194_2421
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3  Reduce: 1   Cumulative CPU: 28.55 sec
Total MapReduce CPU Time Spent: 28 seconds 550 msec
OK
the     142487086
quot    131585830
and     69220506
title   61488231
ref     51751180
amp     49225842
text    49075004
page    43551751
User    39032500
http    31158548
The     30290149
format  30216932
timestamp       29739717
model   29146823
revision        28976784
contributor     28877146
sha     28819730
username        28732925
for     28595464
Category        28259946
comment 28177273
talk    27478425
com     26699461
www     25977261
name    25490183
was     22417581
org     21679663
parentid        20858357
        20676353
date    18870014
that    17894127
with    17732012
style   17065159
space   16651419
from    16130333
wiki    16005668
align   15562284
web     14532426
xml     14515617
preserve        14445312
wikitext        14410739
small   14201593
```

```
minor   6406500
New     6332496
United  6166300
have    6161295
nbsp    6146603
which   6036545
you     5956112
left    5954204
domain  5923750
deletion        5912897
type    5859906
team    5827815
but     5787673
also    5590499
archive 5512341
discussion      5426820
year    5413914
flagicon        5339690
Special 5202424
time    5195381
were    5120157
sup     5077419
other   5028623
one     4964632
class   4929818
WUL     4891112
Time taken: 80.15 seconds, Fetched: 100 row(s)
```

Output of Top-K word (32GB file)

ii) Output of 64GB file

```
hive> select * from final_word_count_64_first_display
    > order by count desc
    > limit 100;
Query ID = bigdata12_20190512223232_6655dcde-60ee-40ae-99f1-309226651977
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
```

```
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 3
Total MapReduce CPU Time Spent: 34 seconds 760 msec
OK
the      281548636
quot     271085749
and      136838428
title    123735761
ref      111849091
text     99382349
page     86694594
amp      83259619
User     76064278
format   61105910
The      59786522
timestamp        58997425
model    58700895
revision         58424690
username         58212679
contributor      58182320
sha      58073343
Category         57342274
for      57340175
comment  56598539
http     56333267
com      55337415
name     53819828
talk     53751334
www      50376680
was      45039361
         44645254
date     41252680
parentid         39373877
style    37672589
org      36110201
that     35303009
with     34877621
align    33993131
from     32951836
space    32583942
wiki     32385619
url      31878758
web      30764208
xml      29224889
preserve         29103817
wikitext         29048235
cite     28550304
small    26620798
UTC      25680193
span     24669608
```

```
team     14729521
news     14223895
diff     13962722
left     13866877
image    13663678
has      13599852
html     13414988
domain   13163006
flagicon         12592406
nbsp     12351719
you      12251849
New      12231320
have     11920128
deletion         11882318
which    11810297
discussion       11526108
United   11499865
oldid    11354677
but      11343834
WUL      11082736
type     11019543
work     11014622
also     11005633
score    10904020
class    10894094
minor    10842971
year     10798170
time     10455893
File     10416546
May      10358260
OtherLinks       10326099
archive  10213975
were     10192131
Time taken: 54.142 seconds, Fetched: 100 row(s)
```

Output of Top-K word (64GB file)

b) Limiting the Length:

-Normal Processing or Display Time Processing: Here the result which we get by creating table at step 2 contains words with all character length so we will limit the result using length() function in the selection query for top 100 words.

```
hive> CREATE TABLE final_word_count_32_first_display AS
    > SELECT word, count(1) AS count FROM
    > (SELECT explode(split(line, '\\s')) AS word FROM raw_32gb_temp ) w
    > GROUP BY w.word
    > ORDER BY w.word;
Query ID = bigdata12_20190512202121_c4e9328e-fb7f-4e32-8a96-6beb56ce3e43
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 514
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1557022631194_2328, Tracking URL = http://name1.hadoop.dc.engr.scu.
Kill Command = /DCNFS/applications/cdh/5.12/app/hadoop-2.6.0-cdh5.12.1/bin/hadoop job
Hadoop job information for Stage-1: number of mappers: 129; number of reducers:
514
2019-05-12 20:22:31,654 Stage-1 map = 0%,  reduce = 0%
2019-05-12 20:23:32,224 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 392.74 sec
2019-05-12 20:23:33,290 Stage-1 map = 1%,  reduce = 0%, Cumulative CPU 539.94 sec
2019-05-12 20:23:34,350 Stage-1 map = 2%,  reduce = 0%, Cumulative CPU 632.14 sec
```

```
MapReduce Total cumulative CPU time: 4 minutes 30 seconds 210 msec
Ended Job = job_1557022631194_2335
Moving data to: hdfs://name1.hadoop.dc.engr.scu.edu:8020/user/hive/warehouse/final_word_count_32_first_display
Table default.final_word_count_32_first_display stats: [numFiles=1, numRows=22626464, totalSize=269019617, rawDataSize=246
MapReduce Jobs Launched:
Stage-Stage-1: Map: 129  Reduce: 514   Cumulative CPU: 10978.86 sec   HDFS Read: 34455498972 HDFS Write: 633729447 SUCCESS
Stage-Stage-2: Map: 18  Reduce: 1   Cumulative CPU: 270.21 sec   HDFS Read: 633908926 HDFS Write: 269019727 SUCCESS
Total MapReduce CPU Time Spent: 0 days 3 hours 7 minutes 29 seconds 70 msec
OK
Time taken: 774.029 seconds
```

```
Stage-Stage-1: Map: 3  Reduce: 1   Cumulative CPU: 49.16 sec
Total MapReduce CPU Time Spent: 49 seconds 160 msec
OK
quot    131585830
title   61488231
text    49075004
page    43551751
User    39032500
http    31158548
format  30216932
timestamp       29739717
model   29146823
revision        28976784
contributor     28877146
username        28732925
Category        28259946
comment 28177273
talk    27478425
name    25490183
parentid        20858357
date    18870014
that    17894127
with    17732012
style   17065159
space   16651419
from    16130333
wiki    16005668
align   15562284
preserve        14445312
wikitext        14410739
small   14201593
cite    12501452
Wikipedia       12328698
span    11633046
wikipedia       11589677
diff    11580424
font    11522225
this    11165250
index   11005113
center  10860384
redirect        10554647
color   10069937
first   10010916
https   9793802
link    9055291
accessdate      8849447
COIBot  8317123
publisher       8040208
oldid   7981094
article 7944509
```

```
December        4206575
Template        4169273
width   4122737
November        4118507
should  4098939
articles        4077369
made    4064664
been    4062590
April   4029750
score   3962634
football        3935515
links   3918356
Time taken: 78.767 seconds, Fetched: 100 row(s)
```

Normal Processing 32GB file

```
hive> CREATE TABLE final_word_count_64_first_display AS
    > SELECT word, count(1) AS count FROM
    > (SELECT explode(split(line, '\\s')) AS word FROM dead123 ) w
    > GROUP BY w.word
    > ORDER BY w.word;
Query ID = bigdata12_20190512212828_da9fedd9-cd68-4b3e-b0e9-82afbd53dfdd
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1027
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1557022631194_2386, Tracking URL = http://name1.hadoop.dc.engr.scu.
Kill Command = /DCNFS/applications/cdh/5.12/app/hadoop-2.6.0-cdh5.12.1/bin/hadoop job
Hadoop job information for Stage-1: number of mappers: 257; number of reducers: 1027
2019-05-12 21:28:41,674 Stage-1 map = 0%,  reduce = 0%
2019-05-12 21:29:06,366 Stage-1 map = 2%,  reduce = 0%, Cumulative CPU 410.86 sec
```

```
2019-05-12 21:50:38,415 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 518.77 sec
MapReduce Total cumulative CPU time: 8 minutes 38 seconds 770 msec
Ended Job = job_1557022631194_2401
Moving data to: hdfs://name1.hadoop.dc.engr.scu.edu:8020/user/hive/warehouse/final_word_count_64_first_display
Table default.final_word_count_64_first_display stats: [numFiles=1, numRows=30227634, totalSize=361175746, rawDataSize=330948112]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 257  Reduce: 1027   Cumulative CPU: 22538.07 sec   HDFS Read: 68931005239 HDFS Write: 847357616 SUCCESS
Stage-Stage-2: Map: 20  Reduce: 1   Cumulative CPU: 518.77 sec   HDFS Read: 847684413 HDFS Write: 361175856 SUCCESS
Total MapReduce CPU Time Spent: 0 days 6 hours 24 minutes 16 seconds 840 msec
OK
Time taken: 1330.635 seconds
hive> select * from final_word_count_64_first_display
    > where length(word) >3
    > order by count desc
    > limit 100;
Query ID = bigdata12_20190512215151_c52334ed-5892-446b-8b11-658f0c768fc1
Total jobs = 1
Launching Job 1 out of 1
```

```
OK
quot     271085749
title    123735761
text     99382349
page     86694594
User     76064278
format   61105910
timestamp        58997425
model    58700895
revision         58424690
username         58212679
contributor      58182320
Category         57342274
comment 56598539
http     56333267
name     53819828
talk     53751334
date     41252680
parentid         39373877
style    37672589
that     35303009
with     34877621
align    33993131
from     32951836
space    32583942
wiki     32385619
preserve         29103817
wikitext         29048235
cite     28550304
small    26620798
span     24669608
wikipedia        24318818
this     21739160
center   21576645
first    21533201
font     21352183
accessdate       21061231
https    20784549
redirect         20483358
color    20287776
link     19089417
publisher        17904347
COIBot   17751367
wikipedia        16431343
article 15767741
last     15276704
background       15235769
index    15199776
team     14729521
```

```
United 11499865
oldid   11354677
type    11019543
work    11014622
also    11005633
score   10904020
class   10894094
minor   10842971
year    10798170
time    10455893
File    10416546
OtherLinks       10326099
archive 10213975
were    10192131
August  10035550
March   9776756
July    9300882
April   9259785
width   9189153
other   9158537
January 9118251
December         9045832
Special 8947655
September        8940071
Template         8789580
October 8689445
November         8603425
June    8459641
American         8430486
County  8349573
made    8328269
been    8323094
should  8219486
football         8189429
States  8109504
February         7782588
user    7742295
about   7704881
right   7593353
list    7493602
Time taken: 242.232 seconds, Fetched: 100 row(s)
```

Normal Processing 64GB file

Optimization:

Method 1: Instead of getting all words into second table at second step we can restrict only those words which have length more than 3. So the data reaching reducers is already restricted.

Method 2: Use the command ‘set hive.exec.parallel=true;’ for trying to do parallel processing. Run the command before loading data to temporary table.

Method 1 Output

```
Time taken: 69.337 seconds, Fetched: 100 row(s)
hive> CREATE TABLE final_word_count_32_first AS
    > SELECT word, count(1) AS count FROM
    > (SELECT explode(split(line, '\\s')) AS word FROM raw_32gb_temp ) w
    > where length(w.word) >3
    > GROUP BY w.word
    > ORDER BY w.word;
Query ID = bigdata12_20190512195454_41b00ea0-f9a3-45f3-824d-7e8eb37a1d62
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 514
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1557022631194_2315, Tracking URL = http://name1.hadoop.dc.engr.scu.edu:
Kill Command = /DCNFS/applications/cdh/5.12/app/hadoop-2.6.0-cdh5.12.1/bin/hadoop job  -ki
Hadoop job information for Stage-1: number of mappers: 129; number of reducers: 514
2019-05-12 19:54:36,828 Stage-1 map = 0%,  reduce = 0%
2019-05-12 19:54:56,748 Stage-1 map = 1%,  reduce = 0%, Cumulative CPU 248.53 sec
2019-05-12 19:54:59,962 Stage-1 map = 2%,  reduce = 0%, Cumulative CPU 504.73 sec
2019-05-12 19:55:03,185 Stage-1 map = 3%,  reduce = 0%, Cumulative CPU 617.57 sec
2019-05-12 19:55:06,396 Stage-1 map = 4%,  reduce = 0%, Cumulative CPU 737.39 sec
2019-05-12 19:55:08,540 Stage-1 map = 6%,  reduce = 0%, Cumulative CPU 801.68 sec
```

32_GB_Method 1

```
hive> CREATE TABLE final_word_count_64_first AS
    > SELECT word, count(1) AS count FROM
    > (SELECT explode(split(line, '\\s')) AS word FROM  dead123) w
    > where length(w.word) >3
    > GROUP BY w.word
    > ORDER BY w.word;
Query ID = bigdata12_20190512220202_87985844-9255-40b1-bd2f-0ae186d9739d
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1027
```

64_GB_Method 1

Method 2: Execution in parallel.

```
2019-05-12 21:04:15,277 Stage-2 map = 100%,  reduce = 97%, Cumulative CPU 525.63 sec
2019-05-12 21:04:20,568 Stage-2 map = 100%,  reduce = 98%, Cumulative CPU 529.79 sec
2019-05-12 21:04:33,271 Stage-2 map = 100%,  reduce = 99%, Cumulative CPU 538.83 sec
2019-05-12 21:04:41,780 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 544.21 sec
MapReduce Total cumulative CPU time: 9 minutes 4 seconds 210 msec
Ended Job = job_1557022631194_2349
Moving data to: hdfs://name1.hadoop.dc.engr.scu.edu:8020/user/hive/warehouse/final_word_count_32_first_parallel
Table default.final_word_count_32_first_parallel stats: [numFiles=1, numRows=22499613, totalSize=268139706, rawDataSize=245640093
MapReduce Jobs Launched:
Stage-Stage-1: Map: 129  Reduce: 514   Cumulative CPU: 13400.06 sec   HDFS Read: 34455764912 HDFS Write: 630846190 SUCCESS
Stage-Stage-2: Map: 19  Reduce: 1   Cumulative CPU: 544.21 sec   HDFS Read: 631027508 HDFS Write: 268139817 SUCCESS
Total MapReduce CPU Time Spent: 0 days 3 hours 52 minutes 24 seconds 270 msec
OK
Time taken: 988.516 seconds
```

32_GB_Parallel

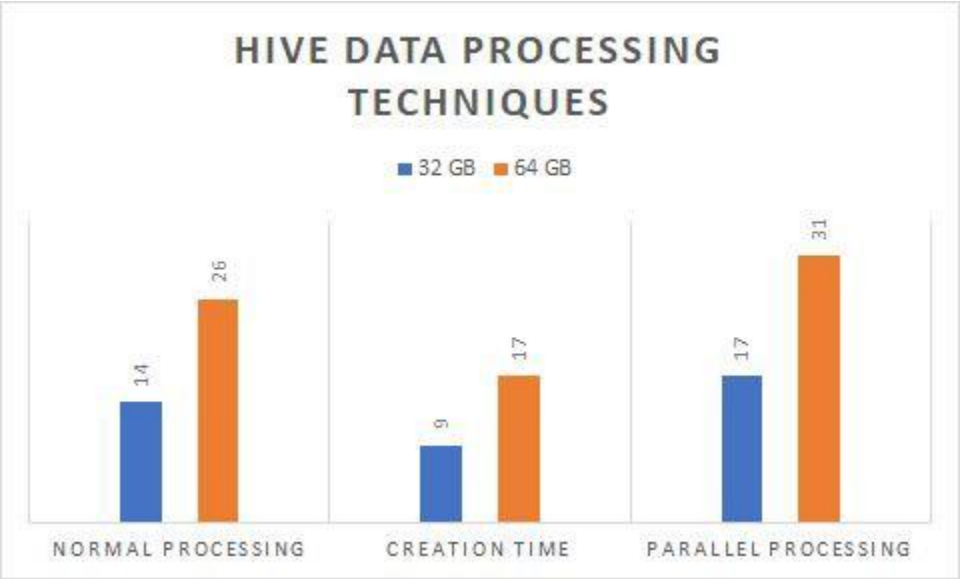
Data visualization

1) Increasing mappers in MapReduce by decreasing default block size.

	32 GB	64 GB
128MB(default)	19	35
64 MB	17	33

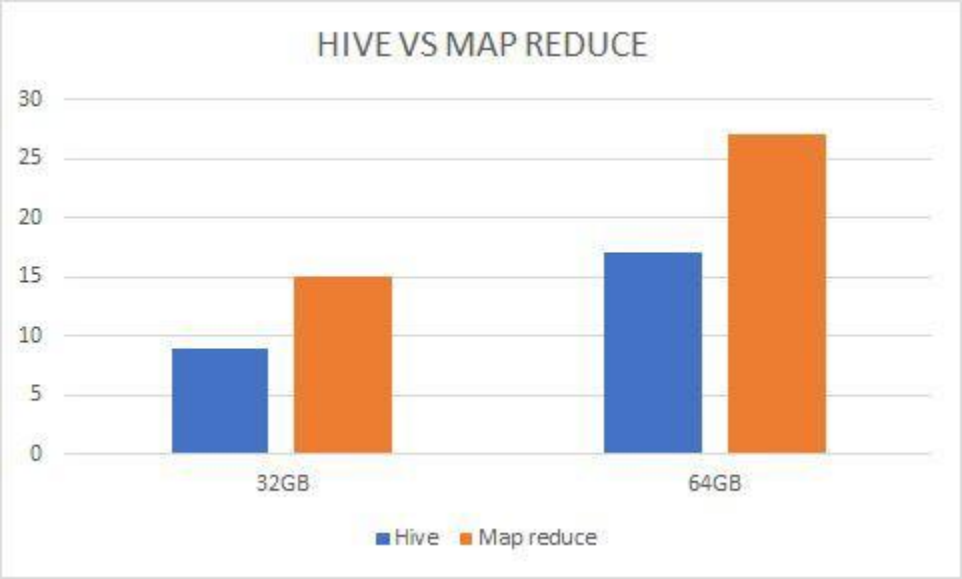
2) Different Methods using hive

	32 GB	64 GB
Display Time Processing	14	26
Creation Time	9	17
Parallel Processing	17	31



3) MapReduce v/s Hive

Platform	32GB	64GB	Method
Hive	9	17	(Creation Time)
Map reduce	15	28	(Approach 1)



Performance Tuning:

- Number of mappers will depend on the dfs block size. 32 GB file was split into 257 blocks. Changing the default block size(128 mb) will affect number of mappers. We changed the default block size from 128 MB to 64 MB which increased number of splits to 515. Increasing number of splits (mappers) had little effect of performance (As shown in chart). Split size can be calculated by below equation.

  ```
  max(mapred.min.split.size, min(mapred.max.split.size,
  dfs.block.size))
  ```

- While working with combiner (which increases the data processed by mapper), we faced problem with default container size (1 GB). The mapperes were processing huge amount of data. While processing 32 GB file, we changed mapreduce.map.memory.mb to 8096 MB. The processing time using combiner was almost similar to the approach 1. (14 minutes). We can also changed the memory allocated to the reducer by mapreduce.reduce.memory.mb. Python program was consuming more memory. Hence, We had to increase the memory allocated to the mapper.
- We also increased number of reducers by changing parameter mapred.reduce.tasks=n. We can pass this value while executing mapreduce command(described in previous section). We processed 32GB file and increased number of reducer tasks, which resulted in taking more time than approach 1. (20 minutes)