Text Analysis using Hadoop MapReduce

Chan-Ching Hsu

The goal is to introduce the MapReduce programming model under Hadoop. MapReduce is a significant abstraction for processing large data sets and has been applied successfully to a number of tasks in the past, including web search. In this exercise, I wrote algorithms for data processing using MapReduce and applied them in analyzing basic statistics of a large text corpus.

Beside this write-up file, in the same directory, you can find output for each experiment in a .txt file, code and executable JAR that produced output.

MapReduce

MapReduce is a programming model for parallel programming on a cluster.

A Program based on the MapReduce model can have several rounds. Each round must have a *map* and a *reduce* method. The input to the program is process by the map method and it emits a sequence of key and value pairs <k, v>. The system groups all values corresponding to a given key and sorts them. Thus for each key, the system generates a list of values corresponding to it, and this is then passed on to the reduce method. The output from the reduce methods can then be used as the input to the next round, or can be output to the distributed file system.

A key point is that two mappers cannot communicate with each other, and neither can reducers communicate with each other. Although this restricts what a program can do, it allows for the system to easily parallelize the actions of the mappers and the reducers. The only communication occurs when the output of a mapper is sent to the reducers.

Resources

A MapReduce program typically has at least 3 classes as shown in the code: A **Driver Class** (in this case, WordCount), a **Map Class** and a **Reduce Class**.

Once can write Java program on local system (using an IDE such as Eclipse). There are

two ways one can compile a Hadoop Map-Reduce program.

1. One can compile the program using Eclipse, while linking the Haddop libraries. Note that Hadoop only needs the jar file to run a MapReduce job, so one can generate the jar file locally and upload it.
2. One can also compile and generate the jar files on cluster. One has to first upload the source files, then do the following:

   ✓ Compile the Java program on the namenode as follows.
   $mkdir class
   $javac –cp "the path to Hadoop jar files" –d class/ WordCount.java
   $jar –cvf WordCount.jar –C class/ . (note the period at the end)

   Or use Eclipse to help compile and export (preferred).

Note

- The MapReduce programs read input from the HDFS and write their output to HDFS. However, one doesn't necessarily need to use the HDFS API while programming with MapReduce.

- Two Hadoop programs running simultaneously cannot have the same output path, although they can share the same input path. Thus, make output path unique, as otherwise job may have a conflict with the output of other jobs, and hence fail. Finally make sure that the output directory is empty by deleting its contents. This must be done every time re-run the program or it will fail.

- Each MapReduce round can have multiple input paths, but only one output path assigned to it. If given an input path that is a directory, MapReduce reads all files within the input directory and processes them.

- The Hadoop MapReduce API: https://hadoop.apache.org/docs/r2.4.1/api/

Experiment

A bigram is a contiguous sequence of two words within the same sentence. For instance, the text "This is a car. This car is fast." has the following bigrams: "this is", "is

a", "a car", "this car", "car is", "is fast". Note that "car this" is not a bigram since these words are not a part of the same sentence. Also note that all upper case letters have been converted to lower case.

Task: Write a Hadoop program to identify the single most frequently occurring bigrams beginning with each letter in an input text corpus, along with their frequencies of occurrence. For example, "ancient history" and "another lambchop" are both bigrams that begin with "a" while "stay shiny" and "serenity firefly" are both bigrams that begin with s. If "stay shiny" occurs 10 times and "serenity firefly" only occurs once, "stay shiny, 10" should be output.

Note that the output format is {letter – bigram, occurrence} and one bigram for one letter.

Output for the Shakespeare dataset

```
[cchsu@n0 output]$ hdfs dfs -cat /scr/cchsu/lab2/exp2/output/part-r-00000
shell-init: error retrieving current directory: getcwd: cannot access parent dir
ectories: No such file or directory
chdir: error retrieving current directory: getcwd: cannot access parent director
ies: No such file or directory
chdir: error retrieving current directory: getcwd: cannot access parent director
ies: No such file or directory
chdir: error retrieving current directory: getcwd: cannot access parent director
ies: No such file or directory
chdir: error retrieving current directory: getcwd: cannot access parent director
ies: No such file or directory
        1 - 1 king,     48
        2 - 2 king,     47
        3 - 3 king,     28
        4 - 4d item,    1
        5 - 5s 8d,      1
        6 - 6d item,    1
        7 - 7 suffolk,  1
        8 - 8d item,    1
        9 - 9 the,      1
        a - and the,    797
        b - by the,     639
        c - come to,    336
        d - do not,     488
        e - enter a,    146
        f - for the,    570
        g - give me,    370
        h - he is,      665
        i - i am,       1889
        j - joan la,    60
        k - king henry, 696
        l - like a,     554
        m - my lord,    1669
        n - no more,    534
        o - of the,     1551
        p - pray you,   379
        q - queen margaret,     153
        r - richard iii,        160
        s - shall be,   554
        t - to the,     1671
        u - upon the,   304
        v - very well,  62
        w - with the,   623
        x - xi then,    3
        y - you are,    717
        z - zounds i,   6
```

OUTPUT FOR THE GUTENBERG DATASET

```
[cchsu@n0 exp2]$ hdfs dfs -cat /scr/cchsu/lab2/exp2/output/part-r-00000
        0 - 00 99,      2325
        1 - 1 e,        7907
        2 - 20 of,      3095
        3 - 30 days,    3098
        4 - 4 of,       2487
        5 - 50 states,  4593
        6 - 60 days,    3075
        7 - 7 and,      754
        8 - 8 1,        562
        9 - 90 days,    3486
        a - and the,    372059
        b - by the,     226489
        c - could not,  61118
        d - did not,    91628
        e - end of,     31824
        f - for the,    230617
        g - going to,   33342
        h - he was,     179027
        i - in the,     749113
        j - just as,    16764
        k - kind of,    23101
        l - like a,     46717
        m - may be,     78373
        n - not to,     46137
        o - of the,     1421985
        p - project gutenberg,  59869
        q - question of,        4860
        r - ready to,   12727
        s - she had,    74870
        t - to the,     564586
        u - upon the,   71980
        v - vb n,       11827
        w - with the,   202607
        x - x 100000000,        555
        y - you are,    63984
        z - zipcorrected editions,      1984
```