ASSIGNMENT 6: MAP REDUCE

PREPARED BY: RAJARAJESWARI VAIDYANATHAN **CWID**: A20362220 **COURSE**: CS422 – DATA MINING

TABLE OF CONTENTS

MAP REDUCE:

MAPPERS:

- Mapper maps input key/value pairs to a set of intermediate key/value pairs.
- Maps are the individual tasks that transform input records into intermediate records. The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs.
- The Hadoop Map/Reduce framework spawns one map task for each InputSplit generated by the InputFormat for the job.

Shuffle / Combiner stage:

- The Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transfer between Map and Reduce.
- Usually, the output of the map task is large and the data transferred to the reduce task is high.

REDUCERS:

- Reducer reduces a set of intermediate values which share a key to a smaller set of values.
- The number of reducers for the job is set by the user via JobConf.setNumReduceTasks (int).
- Technically, Reducer implementations are passed by the JobConf for the job via the JobConfigurable.configure (JobConf) method and can override it to initialize themselves. The framework then calls reduce (WritableComparable, Iterator, OutputCollector, Reporter) method for each <key, (list of values)> pair in the grouped inputs. Applications can then override the Closeable.close () method to perform any required cleanup.
- Reducer has 3 primary phases: shuffle, sort and reduce.

PREPARED BY: RAJARAJESWARI VAIDYANATHAN CWID: A20362220 COURSE: CS422 – DATA MINING

Pseudo code for finding average of the set of integers across different files using Map-reduce.

Please note: The Pseudo-code is highlighted in BLUE for Mapper, Combiner and reducer classes.

Class Mapper

//The files are split by the HDFS and distributed by the HDFS to various Mappers. When it comes to the Mapper function it will emit the key-value pair of the string t (in our case -> Integer) and the number of times it has been reported in the mapper file. In our case since it is an integer and all the integers have to be summed up, Mappers will produce pairs of integers and their count. (though count is constant).

```
Method MAP(string t, integer R)
for all integers R in the file do
    EMIT(string t, pair(R, Count(C) 1))   // Gives the file number along with Integers and counts in pairs.
```

Class Combiner

// When it comes to the combiner(in-map combiner) it will reduce the large amount of data into chunks and compute their average.

```
Method COMBINE(string t, pairs [(R1,C1), (R2,C2), (R3,C3),….(Rn,Cn)]
Integer Sum <- 0           // Initialize the Sum variable.
Integer Count <- 0         // Initialize the Count variable.
Float Cavg                 // Set the data type for Cavg (Computes the average for combiners)
```

// Loop through all pairs of Integers and list of all counts, Compute the sum and Count.

```
for all pairs (R,C) Ɛ pairs [(R1,C1), (R2,C2), (R3,C3),….(Rn,Cn)] do
    Sum   <- sum + R       // Compute the Sum of the integers present in one map file.
    Count <- count + C     // Compute the count of the number of integers present in one map file.
```

// After getting the collection of Count values and the sum values with respect to each file, Compute their average respectively.

```
Cavg   <- Sum / Count
```

// Produce an output with key value pairs of Average (that was computed above and their counts with respect to each files.

```
EMIT(string t, pair(Cavg, Count)]
```

Class Reducer

// When it comes to the reducer, there will be 'n' number of files with their corresponding Average and count values. Reducer function will compute the TotalSum and the totalCount to calculate the average and emit the average to the user for all the integers across various files.

```
    Method REDUCE(string t, pairs [(Cavg1,Count1), (Cavg2,Count2),....(Cavg(n),Count(n))]
    Float Sum <- 0                      // Set the data type of Sum value.
    Float TotalSum <- 0                 // Initialize the TotalSum variable.
    Integer TotalCount <- 0             // Initialize the TotalCount variable.
    Float Ravg                          // Set the data type of Ravg and initialize to zero.
```

// Loop through all pairs of Integers and list of all counts, Compute the sum and Count.

```
    for all pairs (A,C) Ɛ pairs [(Cavg1,Count1), (Cavg2,Count2),....(Cavg(n),Count(n))]
        Sum  <- Cavg * Count            // Compute the Sum. This is calculated with the help of average
and count that was received from Combiners and place the value in Sum (this is done for all the files that are
received from Combiners)

        TotalCount <- Count + TotalCount   // Compute the TotalCount of the number of integers
present in all the map files. Count value is obtained by Combiners with respect to one map file. Store that value in
Totalcount. While reading the 2nd combiner file, again count value is added to the previously stored TotalCount
value and so on and so forth. Likewise, we will get the TotalCounts of all the integers present in all the files
received by the input and stores the final count in TotalCount variable.

        TotalSum <- Sum + TotalSum          // Compute the TotalSum. Since Sum is counted with the
help of average and count received by Combiners, we compute the TotalSum with respect to each files and add
them recursively. Once the Loop ends, we will have an accumulated sum of all the integers (from all the files) and
stored in TotalSum variable.

    Ravg <- TotalSum / TotalCount           // After getting the TotalSum and TotalCount of all the
integers with respect to all the files, we then calculate the average of all the integers across files using the formula
=> TotalSum / TotalCount

    EMIT(Null, float(Ravg)]                 // Produce an output with a Null value (since it has to be
key-value pair) and the average of all the integers across all the files. This value might be in float and that's why
the data type of Average in Reducer is defined as Float.
```

An overflow of how the above Pseudo-code is written.

Mappers Class:

The MAP function takes a series of key/value pairs, processes each and generates output key/value pairs. The input and output of the map are often different from each other.

In our case, Mappers will produce the key value pairs of the File ID and the integers In the file → <File(id), integers count>

Here,

Key = String t (the file number).

Value = Set of integers in each file.

EMIT(String t, pair(R, Count(C) 1)) ===➔ Produces intermediate key-value pairs that are sent to the combiners.

Combiners Class:

Combiner combines the values of all the keys of a single mapper.

For all the key-value pairs, the Combiners, compute the below

Sum = Sum of all integers

Count = Number of integers with respect to one Map file.

Average = Average of the above = Sum / Count (for one Map file)

EMIT(string t, pair(Cavg, Count)] =➔ The output that is produced by the Combiner will be Average and Count of the individual Map file.

Likewise, all the files will emit the pairs of their respective Average and count.

Reducers Class:

Reducer calculates the average of the integers present across all the files.

For all key-value pairs, emitted by the combiners, Reducers will compute the below.

Sum = Average * Count ------------> Step 1

Total Count = Count(This we got from combiner) + TotalCount

Total Sum = Sum of all integers (Calculated in step 1)

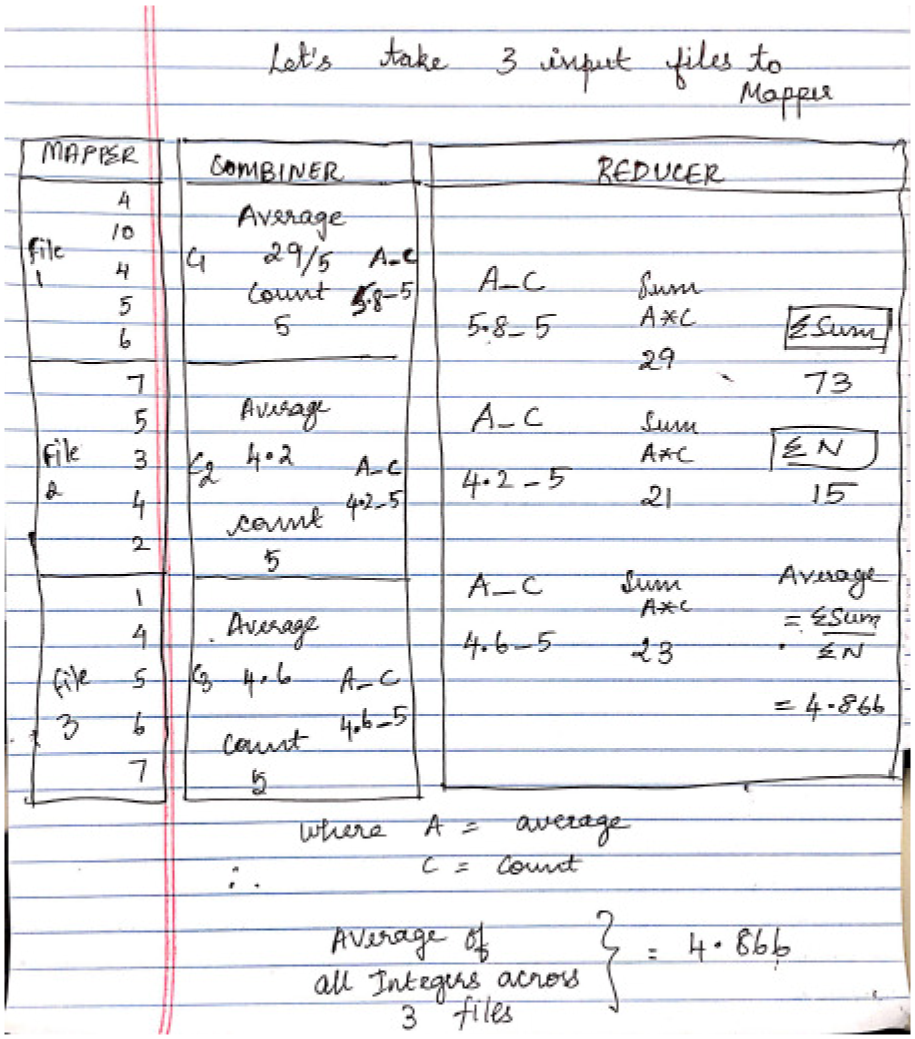
Average = Total Sum / Total Count

EMIT(Null, float(Ravg)] ==➔ Produces the output with the average value of all the integers present in all the map files.

Overall final view of how MapReduce works with respect to the 3 input files with integers.

Here in the Map file 1, though Integer ‘4’ is repeated twice, in-mapper combiner C1 will anyways add all the integers with respect to one mapper file. Likewise it is applicable for all the files. In case of larger files, it is always better to use Combiners to calculate the average or find the max integer.

When using combiners, the calculation will be broken down and it acts more efficiently than the one without combiners where the whole large set of files and integers goes in reducer to do the calculation.



Let's take 3 input files to Mapper

MAPPER		COMBINER		REDUCER		
File 1	4 10 4 5 6	C1 Average 29/5 Count 5	A_C 5.8_5	A_C 5.8_5	Sum A*C 29	$\boxed{\Sigma Sum}$ 73
File 2	7 5 3 4 2	C2 Average 4.2 count 5	A_C 4.2_5	A_C 4.2_5	Sum A*C 21	$\boxed{\Sigma N}$ 15
File 3	1 4 5 6 7	C3 Average 4.6 count 5	A_C 4.6_5	A_C 4.6_5	Sum A*C 23	Average $= \frac{\Sigma Sum}{\Sigma N}$ = 4.866

where A = average
C = Count

$\therefore$ Average of all Integers across 3 files } = 4.866