**BDT – cs523**

**Assignment 4 – Day 4**

---------------------------------------------------------------------------------------------------------------------------------

* Submit your *own work* on time. No credit will be given if the assignment is submitted after the due date.
* Note that the completed assignment should be submitted in .doc, .docx, .rtf or .pdf format only.

**Write an in-mapper combiner algorithm for the “average problem”. Take help from the lecture slides.** (Pseudo code only; show reducer method too.)

ANS:

class Mapper

hashmap<String, pair<int, int>> local\_aggregate;

method map (String u, int t):

if u in local\_aggregate:

pair (sum, cnt) = local\_aggregate.get(u)

sum = sum + t

cnt = cnt + 1

local\_aggregate.put(u, pair(sum, cnt))

else:

local\_aggregate.put(u, pair(t, 1))

// Emit aggregated results for each user after all inputs are processed

method cleanup():

for each (u, pair(sum, cnt)) in local\_aggregate:

Emit(u, pair(sum, cnt))

class Reducer

method reduce (String u, pairs [(s1,c1), (s2,c2),…]):

sum = 0

cnt = 0

for all pair (s, c) in pairs [(s1,c1), (s2,c2),…] do:

sum = sum + s

cnt = cnt + c

avg = sum / cnt

Emit(u, avg)

**Optional - [Following question is for 6 bonus points]**

Assume that there are three reducers. Note that Reducer 1 runs on Machine1. Reducer 2 runs on Machine2. Reducer 3 runs on Machine3.  
Further, let the partitioner assign all words starting from letter ‘a-j’ to Reducer 1, all words starting from letter ‘k-q’ to reducer 2 and everything else to Reducer 3.

Also assume that there are six input splits as follows:

Input split1 : [cherry mango olive cherry]  
 [plum cherry banana cherry]

Input split2 : [cherry banana radish radish]  
 [carrot banana mango cherry]

Input split3 : [banana kiwi plum banana]  
 [mango cherry kiwi banana]

Input split4 : [apple mango carrot plum]  
 [radish kiwi banana olive]

Input split5 : [olive banana radish kiwi]  
 [cherry kiwi olive cherry]

Input split6 : [banana radish plum banana]  
 [olive cherry banana radish]

Input splits 1,2 are on Machine 1, input splits 3,4 are on Machine 2 and input splits 5,6 are on Machine 3.

1. Illustrate the word count algorithm with combiner, no in-mapper combining. (assume that the combiner will work all the time)  
   *show mapper o/p, combiner o/p, reducer i/p and reducer o/p*
2. Illustrate the word count algorithm with in mapper combiner.  
   *show mapper o/p, reducer i/p and reducer o/p*

Remember to show the sorted mapper output that gets stored locally.   
*Note: Illustrate means show mapper o/p, combiner o/p (if using combiners), reducer i/p and reducer o/p.*

**Answers:**

1. With combiner, no in mapper combining (assume that the combiner will work all the time)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Machine 1** | | **Machine 2** | | **Machine 3** | |
| **Mapper 1 output for Input Split 1** | | **Mapper 3 output for Input Split 3** | | **Mapper 5 output for Input Split 5** | |
| <cherry, 1>  <mango, 1>  <olive, 1>  <cherry, 1> | <plum, 1>  <cherry, 1>  <banana, 1>  <cherry, 1> | <banana, 1>  <kiwi, 1>  <plum, 1>  <banana, 1> | <mango, 1>  <cherry, 1>  <kiwi, 1>  <banana,1> | <olive, 1>  <banana, 1>  <radish, 1>  <kiwi, 1> | <cherry, 1>  <kiwi, 1>  <olive, 1>  <cherry, 1> |
|  | |  | |  | |
| **Combiner 1 output– saved locally as mapper1 output file** | | **Combiner 3 output– saved locally as mapper3 output file** | | **Combiner 5 output– saved locally as mapper 5 output file** | |
| <banana, 1>  <cherry, 4>  <mango, 1>  <olive, 1>  <plum, 1> | | <banana, 3>  <cherry, 1>  <kiwi, 2>  <mango, 1>  <plum, 1> | | <banana, 1>  <cherry, 2>  <kiwi, 2>  <olive, 2>  <radish, 1> | |
|  | |  | |  | |
| **Mapper 2 output for Input Split 2** | | **Mapper 4 output for Input Split 4** | | **Mapper 6 output for Input Split 6** | |
| <cherry, 1>  <banana, 1>  <radish, 1>  <radish, 1> | <carrot, 1>  <banana, 1>  <mango, 1>  <cherry, 1> | <apple, 1>  <mango, 1>  <carrot, 1>  <plum, 1> | <radish, 1>  <kiwi, 1>  <banana,1>  <olive, 1> | <banana, 1>  <radish, 1>  <plum, 1>  <banana, 1> | <olive, 1>  <cherry, 1>  <banana, 1>  <radish, 1> |
|  | |  | |  | |
| **Combiner 2 output– saved locally as mapper2 output file** | | **Combiner 4 output– saved locally as mapper4 output file** | | **Combiner 6 output– saved locally as mapper6 output file** | |
| <banana, 2>  <carrot, 1>  <cherry, 2>  <mango, 1>  <radish, 2> | | <apple, 1>  <banana, 1>  <carrot, 1>  <kiwi, 1>  <mango, 1>  <olive, 1>  <plum, 1>  <radish, 1> | | <banana, 3>  <cherry, 1>  <olive, 1>  <plum, 1>  <radish, 2> | |
| **Shuffle & Sort** | | | | | |
| **Reducer 1 input** | | **Reducer 2 input** | | **Reducer 3 input** | |
| <apple, 1>  <banana, [1,2,3,1,1,3]>  <carrot, [1,1]>  <cherry, [4,2,1,2,1]> | | <kiwi, [2,1,2]>  <mango, [1,1,1,1]>  <olive, [1,1,2,1]>  <plum, [1,1,1,1]> | | <radish, [2,1,1,2]> | |

Reducer output is the same for both the cases:

|  |  |  |
| --- | --- | --- |
| **Reducer 1 output** | **Reducer 2 output** | **Reducer 3 output** |
| Apple **1**  Banana **11**  Carrot **2**  Cherry **10** | Kiwi **5**  Mango **4**  Olive **5**  Plum **4** | Radish **6** |

1. With in-mapper combining

|  |  |  |
| --- | --- | --- |
| **Machine 1** | **Machine 2** | **Machine 3** |
| **Mapper 1 output for Input Split 1** | **Mapper 3 output for Input Split 3** | **Mapper 5 output for Input Split 5** |
| <cherry, 4>  <mango, 1>  <olive, 1>  <plum, 1>  <banana, 1> | <banana, 3>  <kiwi, 2>  <plum, 1>  <mango, 1>  <cherry, 1> | <olive, 2>  <banana, 1>  <radish, 1>  <kiwi, 2>  <cherry, 2> |
|  |  |  |
| **Mapper 2 output for Input Split 2** | **Mapper 4 output for Input Split 4** | **Mapper 6 output for Input Split 6** |
| <cherry, 2>  <banana, 2>  <radish, 2>  <carrot, 1>  <mango, 1> | <apple, 1>  <mango, 1>  <carrot, 1>  <plum, 1>  <radish, 1>  <kiwi, 1>  <banana, 1>  <olive, 1> | <banana, 3>  <radish, 2>  <plum, 1>  <olive, 1>  <cherry, 1> |
| **Shuffle & Sort** | | |
| **Reducer 1 input** | **Reducer 2 input** | **Reducer 3 input** |
| <apple, [1]>  <banana, [1, 2, 3, 1, 1, 3]>  <carrot, [1, 1]> <cherry, [4, 2, 1, 2, 1]> | <kiwi, [2,1,2]>  <mango, [1,1,1,1]>  <olive, [1,1,2,1]>  <plum, [1,1,1,1]> | <radish, [2,1,1,2]> |