Distributed Data Processing and Analysis (BIG DATA) (CENG595) - Erdoğan Doğdu
Spring 2017 – Midterm – Çankaya University
Apr 25, 2017 – 90 min **Student Number**: **Name**:

Question	***1s***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***Total***
Value	5	10	10	10	25	15	10	15	*100*
Points									

1. (5p) Which strategy below best explains MapReduce approach for distributed data processing?
   a. Heuristic Algorithm
   b. Divide and Conquer
   c. Dynamic Programming
   d. Game Theory
   e. None of the above

2. (5p) Order the following steps (1 to 5) that constitute the MapReduce algorithm. Also write if the step is part of Map (M) or Reduce (R):
   (_1__) ___M___ Iterate over a large number of records
   (_2__) ___M___ Extract something of interest from each
   (_3__) _____ Shuffle and sort intermediate results
   (_4__) ___R___ Aggregate intermediate results
   (_5__) ___R___ Generate final output

3. (5p) Which tasks below are handled by the MapReduce runtime environment? Check <u>all</u> that apply.
   __x__ Handles scheduling: Assigns workers to map and reduce tasks
   __x__ Handles "data distribution": Moves processes to data
   __x__ Handles synchronization: Gathers, sorts, and shuffles intermediate data
   _____ Handles combining: Automatically detects opportunities for combining in mapper and executes
   __x__ Handles errors and faults: Detects worker failures and restarts

4. (5p) How can you reduce communication between nodes of a MapReduce cluster? Check all that apply:
   __x__ Adding in-mapper combining
   __x__ Adding a combiner between map and reduce
   _____ Adding a partitioner
   _____ Adding more nodes to the cluster

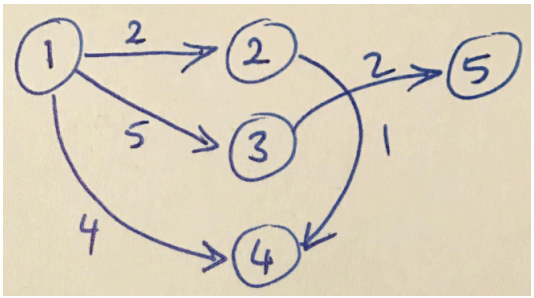
5. (25p) Consider the assignment 2 where the data file consists of directed graph edges of the form:

   source_url, destination_url, weight

   a) (5p) Below is a list of sample lines from the data file:

   1, 2, 2
   1, 3, 5
   1, 4, 4
   2, 4, 1
   3, 5, 2

   Draw the graph with the nodes as circles and edges as directed arrows from node to node with weights.



b) (10p) Write a MapReduce algorithm to find the *in degree* (number of incoming edges) of the nodes in the graph. For the above example, the algorithm for example will output the following lines (*node#, in-degree*):

2, 1
3, 1
4, 2
5, 1

Solution:

```
map(k,v) {
  k'=v.split(',')[1]  // destination_url
  v'=1
  emit(k',v')
}

reduce(k,v[]) {
  // input: k:des_url, v=[1,1,...]
  k'=k
  sum=0
  for each x in v[]
    sum += x
  v'=sum
  emit(k',v')
}
```

c) (10p) Improve the algorithm in (b) with in-mapper combining and combine functions.

Solution:

```
map(k,v) {  // with in-mapper combining
  initialize:
    H = new AssociativeArray()
  method:
    d=v.split(',')[1]  // destination_url
    H{d} = H{d} + 1
  close:
    for all t in H:
      emit(t, H{t})
}

combine(k,v[]) {  // combine, same as reduce
  // input: k:des_url, v=[4,2,1,2,...]
  k'=k
  sum=0
  for each x in v[]
    sum += x
  v'=sum
  emit(k',v')
}
```

```
reduce(k,v[]) {
  // input: k:des_url, v=[4,2,1,2,...]
  k'=k
  sum=0
  for each x in v[]
    sum += x
  v'=sum
  emit(k',v')
}
```

6. (HDFS, 15p) Briefly answer the following questions about HDFS:
   a. What is HDFS short for? (full name)

      Hadoop Distributed File System_____

   b. What is the size of a data block in HDFS? _____**64MB or 128MB**_____

   c. How many copies of each data block are stored? _____**3**_____

   d. How many racks and how many servers are used to store the copies of a data block?

      2 racks and 3 servers in total (2 servers in one rack and 1 server in another rack)

   e. What is the reason for storing data block copies in separate racks and servers?

      The reason for saving two copies in the two separate servers on the same rack is to decrease the communication time; communication time between racks is longer than communication time between two servers on the same rack. The reason for storing multiple copies in multiple servers is to be able to recover from error; in case a copy is lost it can be copied from one of the other two copies (either, possibly, from the same rack, or from another rack in the worst case).

7. (Hadoop, 10p) Explain *datanode*, *namenode*, *tasktracker*, *jobtracker* in the context of Hadoop. What does each one do? How many of each exists in a typical installation?

Solution:

These are servers in a Hadoop installation. Below are their responsibilities:

Namenode: There is one namenode server. It keeps track of datanode servers and data blocks, and also keeps track of which datanode server holds one of the 3 copies of each data block. HDFS operations are coordinated with namenode server.

Datanode: All servers except the namenode are datanode servers. They store data blocks (3 copies each). They serve data to mapreduce jobs (read).

Jobtracker: There is one jobtracker server. It monitors all mapreduce jobs. Jobs are first sent to jobtracker node. Jobtracker server creates map and reduce tasks and assigns them to the relevant tasktracker servers.

Tasktracker: All servers except the jobtracker are tasktracker nodes. They execute map, reduce, and combine tasks which are assigned by the jobtracker node.

8. (15p) Consider the following data file with the format: *Year, Month, Day, Temperature*

```
2012, 01, 01, 5
2012, 01, 02, 45
2012, 01, 03, 35
2012, 01, 04, 10
...
2001, 11, 01, 46
2001, 11, 02, 47
2001, 11, 03, 48
2001, 11, 04, 40
...
2005, 08, 20, 50
2005, 08, 21, 52
2005, 08, 22, 38
2005, 08, 23, 70
```

And, suppose you want to get the following output using a MapReduce algorithm. That is the in format of Year-Month: Temp1, Temp2, Temp3, …

```
2001-11: 40, 46, 47, 48, ...
2005-08: 38, 50, 52, 70, ...
2012-01:  5, 10, 35, 45, ...
```

Notice that the year-month (key) is increasing and the temperatures in each month (values) are also in increasing order.

a) Which design pattern would you use to get an ordered list of values for each key?
b) Write the mapreduce program to get this output.

Solution:

a) Secondary sorting
b) Needs partitioner:

```
map(k,v):
  val = v.split(‘,’)
  k’= val[0]+”-“+val[1]+”-“+val[3]
  v’= “”
  emit(k’,v’)

partitioner(k):
  return hashkey(k.substr(0,7)) % numPartitions

reduce():
  initialize:
    k’ = k
    v’ = “”
  method:
```

```
  if (k != k') {
    emit (k',v')
    k' = k
  }
  for each x in v[]
    v' = v' + " " + x
close:
  emit (k',v')
```