Birkbeck

(University of London)

BSc/FD EXAMINATION

Department of Computer Science and Information Systems

Cloud Computing Concepts (BUCI028H6)

CREDIT VALUE: 15 credits

Date of examination: Friday, 24th May 2013
Duration of paper: 2:30pm – 4:30pm (2 hours)

RUBRIC

1. *This paper contains* 6 *questions for a total of* 100 *marks.*
2. *Students should attempt to answer* **all** *of them.*
3. *This paper is not prior-disclosed.*
4. *The use of non-programmable electronic calculators is permitted.*

1. **(30 marks)**

Please give brief answers to the following questions.

(a) What are the five principles of cloud computing? (5 marks)

(b) What do IaaS, PaaS, and SaaS stand for respectively? Provide an example for each one of them. (6 marks)

(c) What is the main economic benefit of cloud computing? (2 marks)

(d) In a RESTful architecture, what HTTP method should be used to replace a collection or element with a new one? Is that method idempotent or not? (2 marks)

(e) What does NoSQL mean? Provide an example of NoSQL. (3 marks)

(f) What does hybrid cloud mean? When should an organisation consider using a hybrid cloud? (3 marks)

(g) What is the CAP theorem? (5 marks)

(h) What is the major advantage of consistent hashing over traditional hashing? (2 marks)

(i) Name two distributed computing frameworks that were inspired by the Bulk Synchronous Parallel (BSP) model. (2 marks)

2. **(20 marks)**

Suppose we execute the following word-count MapReduce program on a large repository (such as a copy of Wikipedia), using 100 Map tasks and some number of Reduce tasks.

```
class MAPPER
    method MAP(docid $a$, doc $d$)
        for all term $t \in$ doc $d$ do
            EMIT(term $t$, count $1$)
class REDUCER
    method REDUCE(term $t$, counts $[c_1, c_2, \ldots]$)
        $sum \leftarrow 0$
        for all count $c \in$ counts $[c_1, c_2, \ldots]$ do
            $sum \leftarrow sum + c$
        EMIT(term $t$, count $sum$)
```

(a) If we do *not* use a combiner at the Map tasks, and make each reducer a separate Reduce task, do you anticipate a significant skew (say more than 1,000x difference) in the amount of time taken by each Reduce task or not? Explain your answer. (3 marks)

(b) If we do *not* use a combiner at the Map tasks, and combine the reducers into 10 Reduce tasks at random, do you anticipate a significant skew or not? Explain your answer. (3 marks)

(c) Can we use the reducer function directly as the combiner function? Explain your answer. (3 marks)

(d) If we do use a combiner at the Map tasks, and combine the reducers into 10,000 Reduce tasks at random, do you anticipate a significant skew or not? Explain your answer. (3 marks)

(e) How can one re-write the above MAPPER class using the "in-mapper combining" design pattern? (4 marks)

(f) What are the advantages and drawbacks of using the "in-mapper combining" design pattern instead of a combiner? (4 marks)

3. **(15 marks)**

Suppose we have a very large file of integers (split over a number of machines) as the input. Note that a single machine does not have sufficient capacity to hold all the distinct integers from the input. Please design MapReduce algorithms for the following tasks. You should try to use combiners (not "in-mapper combining") whenever possible to optimise the program's performance.

(a) Obtain the largest integer. (3 marks)

(b) Obtain the average of all integers. (4 marks)

(c) Obtain the number of distinct integers. (8 marks)

4. **(10 marks)**

Consider a MapReduce program to compute "host sizes". The input file has records of the form $\langle$`URL, size`$\rangle$, and you are given a function `url_host(URL)` that, given a URL, returns the hostname. The goal is to compute the sum of the sizes of all the web pages for each hostname in the input file. Suppose we use MapReduce with 10 map workers and 5 reduce workers. Assume that the map reduce implementation schedules the map tasks on the nodes that contain chunks of the input file on their local disks, and that no two tasks (map/map, map/reduce, reduce/reduce) are scheduled on the same node. The distributed files system (HDFS) uses 2-way replication. You can assume that one replica of each chunk will be written on the same node as the task that produces it. The input file contains 100 million records, representing 10 million unique hosts, and each chunk contains on average 3 million unique hostnames. The average length of a URL is 100 bytes, the average length of a hostname is 20 bytes, and sizes are encoded using 4-byte integers.

(a) Assume that we use a MapReduce implementation without a combiner.
Estimate the total disk I/O and network I/O during the computation. (5 marks)

(b) Assume that we use a combiner to improve the performance of our program.
Estimate the total disk I/O and network I/O during the computation. (5 marks)

5. **(10 marks)**

Calculate the communication cost between the mappers and the reducers, i.e., the total number of (key, value) pairs that are sent from the mappers to the reducers, for each of the following problems. Assume that there is no combiner or "in-mapper combining". Justify your answer.

(a) Compute the multiplication of two matrices — one matrix A of size $l \times m$ and the other matrix B of size $m \times n$ — in one MapReduce step, with each reducer computing the value of a single element at entry (i, k) (where $i \in [1, l]$, $k \in [1, n]$) in the matrix product. (5 marks)

(b) Compute the cross product of two sets — one set A of size m and the other set B of size n — in one MapReduce step, with each reducer handling all the items in the cross product corresponding to a single item $\in A$. As an example, the cross product of two sets $A = \{u, v\}$, $B = \{x, y, z\}$ is $\{(u, x), (u, y), (u, z), (v, x), (v, y), (v, z)\}$, so there will be one reducer generating $\{(u, x), (u, y), (u, z)\}$, and the other generating $\{(v, x), (v, y), (v, z)\}$. (5 marks)

6. **(15 marks)**

Please write a MapReduce program (in pseudo-code) that implements a simple "People You Might Know" social network friendship recommendation algorithm. The key idea is as follows: if two people have many mutual friends, the system recommends that they connect to each other.

Input:

The input file contains the adjacency list and has multiple lines in the following format:

`<User><TAB><Friends>`

where `<User>` is a unique integer ID corresponding to a unique user, and `<Friends>` is a comma separated list of unique IDs corresponding to the friends of that user. Note that the friendships are mutual (i.e., edges are undirected): if a is a friend of b then b is also a friend of a.

Algorithm:

Let us use a simple algorithm such that, for each user u, the algorithm recommends 10 users who are not already friends with u, but have the most number of mutual friends in common with u.

Output:

The output should contain one line per user in the following format:

`<User><TAB><Recommendations>`

where `<User>` is a unique ID corresponding to a user and `<Recommendations>` is a comma separated list of unique IDs corresponding to the algorithm's recommendation of people that `<User>` might know, in the descending order of the number of mutual friends.

(a) For each user u, your program can get the list of people who have mutual friends with u. (5 marks)

(b) For each user u, your program can get the list of people who have mutual friends with u in the descending order of the number of mutual friends. (5 marks)

(c) For each user u, your program can exclude the existing friends of u from the list of people for friendship recommendation. (3 marks)

(d) For a user u with no existing friend in the social network, your program can output an empty list. (2 marks)