香 港 中 文 大 學
The Chinese University of Hong Kong

Course Examinations 2018-19 (1st term)

Course Code & Title : **CSCI4180 Introduction to Cloud Computing and Storage**

Time allowed : 2 **hours** 0 **minutes**

Student I.D. No. : ..... **Seat No. :** .....

You have two hours to complete the exam. All questions are to be completed. The full score is **100 points**. You are allowed to bring two A4-sized cheat sheets, with both sides printed or written, and use an electronic calculator approved by the University. Other electronic equipments are prohibited. Write down your **student ID** and **seat number** in the answer book. Write **all** the answers in the answer book. Write **neatly**. Anything that is unreadable will receive zero point.

Questions

1. **(24%, 4% each) Quick Questions.** Keep your answers short and precise.

   (a) Name two differences between public clouds and traditional data centers.

   (b) HDFS requires that the block size is sufficiently large. State one advantage and one disadvantage for choosing a large block size.

   (c) Under the CAP algorithm, if we ensure that both C and P hold (e.g., in distributed databases), we cannot ensure that A holds. Explain why.

   (d) Instead of using simple hash functions (e.g., SHA256(x) mod n, where x is the data item and n is the number of server), Amazon Dynamo uses consistent hashing. Explain the reasons.

   (e) Explain what "atomic" means in the data access of Zookeeper.

   (f) State two possible causes that lead to the tail latency problem in distributed storage systems.

2. **(16%) MapReduce.**

   Suppose that we are given the employee records, each of which is represented as a string `record` = "name, age, country, salary". Each record is identified by the employee ID (denoted by an integer `employeeID`). Our goal is to compute the *variance* of the salaries of all employees for each age in each country. We use MapReduce to solve this problem.

   Hint: For a sequence of numbers $\langle x_1, x_2, \cdots, x_n \rangle$, we can compute the variance as follows

   $$v = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}(\sum_{i=1}^{n} x_i^2) - \bar{x}^2,$$

   where $\bar{x}$ is the mean of the sequence of numbers.

   We use *in-mapper combining* to improve the performance.

   Write down the pseudo-code of the map and reduce functions, as well as other necessary functions for realizing in-mapper combining in the mapper class, based on the following map and reduce interfaces:

   - map (**Integer** `employeeID`, **String** `record`)
   - reduce (**Type1** k, **Type2** [v1, v2, $\cdots$])

   You'll need to decide the correct data types in the reduce function. For your reference, Figure 1 shows the MapReduce pseudo-code of the Mean Computation Version 1 in Lecture 4.

Go to Next Page...

```
1: class Mapper
2:     method Map(string t, integer r)
3:         Emit(string t, integer r)

1: class Reducer
2:     method Reduce(string t, integers [r_1, r_2, ...])
3:         sum ← 0
4:         cnt ← 0
5:         for all integer r ∈ integers [r_1, r_2, ...] do
6:             sum ← sum + r
7:             cnt ← cnt + 1
8:         r_avg ← sum/cnt
9:         Emit(string t, integer r_avg)
```

Figure 1: MapReduce pseudo-code of Mean Computation Version 1 for Question 2.

3. **(12%) Shortest-path Algorithm.**

Consider the network shown in Figure 2. Using the parallel Dijkstra's algorithm based on MapReduce, show how you can solve for the shortest path distance from node n_0 to each of other nodes. You may define N_i (where $0 \leq i \leq 3$) as the node structure of node n_i that contains the adjacency list of node n_i and the current shortest path distance from node n_0. In your answers, you need to show (i) the initialized shortest path distances of all node structures, (ii) the outputs emitted by the Map function, and (iii) the shortest path distances emitted by the Reduce function. You only need to show the *first two* MapReduce iterations.

Note: all edges are undirected.

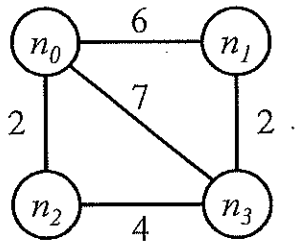

Figure 2: Network topology for Question 3.

4. **(28%) Cloud Storage and Deduplication.**

We develop a deduplication system for cloud storage. We use Rabin fingerprinting as the chunking algorithm. Let us assume that we store a file with the following byte sequence $\{t_1, t_2, \cdots\}$, where t_i denotes the i^{th} byte. Then we operate on the byte sequence and generate Rabin fingerprints (RFPs) based on the following formula:

$$p_s(d,q) = \begin{cases} \left(\sum_{i=1}^{m} t_i \times d^{m-i}\right) \bmod q, & s = 0 \\ \left(d \times \left(p_{s-1} - d^{m-1} \times t_s\right) + t_{s+m}\right) \bmod q, & s > 0 \end{cases}$$

for some parameters m, q, and d.

(a) (4%) Suppose that we set $q = 2^k$ for some integer k, and denote each RFP by p. We generate anchor points (i.e., chunk boundaries) using the following pseudo-code:

```
if ((p & q) == 0) {
   put one anchor point;
}
```

Unfortunately, there is a bug in the above pseudo-code. Please write down the correct pseudo-code for the anchor point generation, while ensuring that the computational efficiency is maintained as in the above (wrong) pseudo-code.

(b) (4%) Based on the corrected pseudo-code in Part (a), how many anchor points are generated on average for storing a file of size F? Explain your answer. State any of your assumptions.

(c) (4%) Based on the corrected pseudo-code in Part (a), explain a type of workload that generates the *most* anchor points. How many anchor points are generated in this case for storing a file of size F?

(d) (4%) We now put the full fingerprint index on disk and deploy a Bloom filter to save disk I/O. Suppose that the Bloom filter is configured with a false positive probability 0.01. Also, consider a workload with M chunks before deduplication and the deduplication ratio is 5:1. Derive the expected number of disk I/O for accessing the fingerprint index. State any of your assumptions.

(e) (4%) Explain how offline (or out-of-order) deduplication solves the fragmentation problem of the most recently stored file.

(f) (4%) Practical deduplication systems group chunks into *containers*, which form the basic read/write units. State one advantage and one disadvantage of this design.

(g) (4%) To protect convergent encryption against the offline brute-force attack, one solution is to encrypt the chunks with the key $h(s, M)$, where s is the file owner's secret, M is the chunk content, and h is the cryptographic hash function. Explain whether it can protect against the offline brute-force attack. How is the overall deduplication effectiveness affected with this approach?

5. **(20%) Facebook's photo storage.**

(a) (4%) Explain why Haystack stores photos in a log-structured manner.

(b) (9%) Recall that f4 is originally configured with $(n, k) = (14, 10)$ Reed-Solomon codes in each cell to tolerate four disk failures. Suppose that it now changes the configuration to $(n, k) = (16, 12)$. Explain how it affects storage redundancy, single-block repair performance, and single-block update performance *quantitatively*, in terms of the percentage of increase/decrease.

(c) (3%) Suppose that f4 uses $(n, k) = (16, 12)$ within a cell and further uses $(n, k) = (4, 2)$ Reed Solomon coding for four cells. What is the redundancy factor?

(d) (4%) Based on Part (c), what is the minimum number of failed disks such that the data loss is irrecoverable? Explain your answers.

Hope you have a fruitful winter break!

-End-