BIG DATA MANAGEMENT

CZ/CE4123

Tutorial 6
More Practice for quiz

MAIN TAKE-AWAY MESSAGE

1. Procedures/reasons are important.
2. Consider all cases.
3. Just memorizing the knowledge is not perfect.

QUESTION1

Given the following three tables (primary keys are underlined):

Employee(<u>EID</u>, Salary)
Manager(<u>MID</u>, Salary)
Employee-Manager(<u>EID</u>, MID)

Each manager supervises at least one employees. Employee-Manager is a table that contains the manager ID (i.e., MID) for each employee (i.e., EID). How to convert the relational data model to a key-value data model? Consider that the main purpose of the conversion is for the query “Given an employee ID, find the salary of the employee’s manager”. The conversion should retain the information as much as possible.

Step 1: Join Employee and Employee-Manager on attribute EID

Step 2: further (Left) join Manager on attribute MID to get Table

(EID, MID, Manager-Salary, Employee-Salary)

Given the main purpose, we set

key as EID;

As the conversion needs to keep the information as much as possible, we set

value as MID;Manager-Salary;Employee-Salary;

COMMON MISTAKES

1. Only give the answer without procedures;
2. Incorrectly set the key to be EID;MID;
3. Only retain part of the information;

QUESTION 2

Consider reading data from memory hierarchy consisting of L1 Cache, L2 Cache, and main memory with the following parameters.

L1 Cache:

- Read access time: 2 nanoseconds
- Miss ratio: 0.4

L2 Cache:

- Read access time: 10 nanoseconds
- Miss ratio: 0.2

Main memory:

- Read access time: 100 nanoseconds.

Estimate the average data read cost and explain your answer. (Note: consider L1, L2 caches and main memory only).

SOLUTION

First access L(i), having a cost c1;

Then, if there is an L(i) miss, we need to access L(i+1). This happens with Probability m1 (miss rate) and a cost of c2.

Then, if there is an L(i+1) miss, we need to access L(i+2). This happens with probability m1m2 (both L(i) and L(i+1) miss) and a cost of c3.

Average cost = **c1+m1c2+m1m2c3 = 2+0.4*10+0.4*0.2*100=14 nanoseconds.**

ALTERNATIVE SOLUTION

If L1 cache hit,

probability = 1-0.4=0.6

If L1 cache miss and L2 cache hit,

probability = 0.4*(1-0.2)=0.32

If L1,L2 caches miss (data in memory),

probability = 0.4*0.2=0.08

Average data read cost = 2*0.6 + (2+10)*0.32+(2+10+100)*0.08=14ns

QUESTION 3

In the lecture, we introduced a cost-free magic function telling which pages locate the qualified data for a query. Consider a disk page size is 1024 integers. There are 10240 integers, which are 1, 2, 3, …, 10240, sequentially stored at 10 consecutive disk pages. Consider the following 4 queries using 4 scans over the data, where each query range is decided by three integers x, y, z, and $1<x<y<z<10240$.

Query 1: searching values in the range [1, x] (i.e., values at least 1 and at most x)
Query 2: searching values in the range [x+1, y]
Query 3: searching values in the range [y+1, z]
Query 4: searching values in the range [z+1, 10240]

List **all possible** total number of read I/Os needed for the 4 scans, *with* the magic function. Please explain your answer.

QUESTION 5

Based on the magic function, the total #pages of 4 searches is roughly 10 (but with possible additional cost) because the 4 ranges are disjoint and covering all the range [1, 10240].

The best case happens when x, y, z are in the exact page boundaries (right boundary), i.e., x and x+1 are stored at different pages; y and y+1 are stored at different pages; z and z+1 are stored at different pages. In this case, total I/Os=10.

Also note that, if x is in the middle of a page, i.e., x and x+1 are in the same page, then this page is accessed by 2 searches.

Hence, the worst case is 10+3 = 13pages, where x, y, z are in the middle.

Finally, easy to see 10, 11, 12, 13 are all possible, because

(1) Incurring 11 page reads happens when only one of (x,y,z) is in the middle of a page.

(2) Incurring 12 page reads happens when only two of (x,y,z) are in the middle of a page.

QUESTION 4

We have a 32-integer array *A* in the main memory. Let cache size be 16 (integers), and cache line size be 4 (integers). Suppose that initially the cache is empty, and the cache replacement policy is the same as the one introduced in the lectures, i.e., first cached first evicted. Let the *f-trip* and *b-trip* scanning be the following.

```
f-trip(){
    for (int j=0; j<32; j++){
        Access A[j];// Access does not change the data
    }
}
b-trip(){
    for (int j=0; j<32; j++){
        Access A[31-j];
    }
}
```

If we need to do 99 scans of the array, and we can select each scan to be either f-trip or b-trip. Please give one best selection strategy that gives the minimum number of misses and explain your answer. Please also compute the number of cache hits and cache misses in the best strategy.

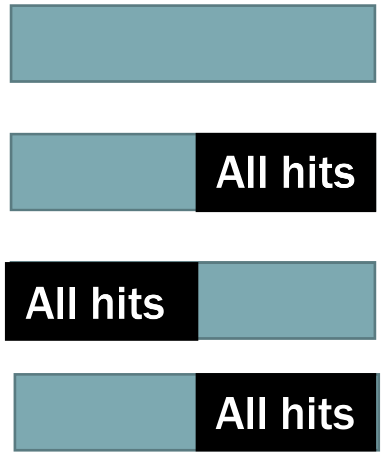
QUESTION 2

The best strategy:

Take turns to use f-trip and b-trip, namely,

f-trip, b-trip, f-trip, …

The idea is that, starting from the 2nd trip, we can always make use of the cache from the previous trip.



Number of hits & misses

1st f-trip: one miss brings three hits. So there are 8 misses and 24 hits.

The remaining 98 trips: first 16 accesses are hits; the last 16 accesses also have the pattern – one miss brings three hits. In total: 28 hits and 4 misses.

Total hits: 24+28*98=2768

Total misses: 8+4*98=400