15-440/15-640: Distributed Systems

Homework 2

REFERENCE SOLUTION

Last update: February 24, 2016

Question 1

Consider a LRU page cache that receives the following reference stream (the page number of each reference is shown):

$$27, 12, 15, 15, 15, 27, 34, 15, 12, 27, 34, 15, 15, 15, 12, 8$$

Suppose the cache size is 3 pages, and the cache is initially empty.

Solution: Based on the access stream, and LRU cache should have the following states after each step:

Step	Current access	Cache (most recent → least recent)	Miss count
0	-	$\varnothing$	0
1	27	27	1
2	12	12 → 27	2
3	15	15 → 12 → 27	3
4	15	15 → 12 → 27	3
5	15	15 → 12 → 27	3
6	27	27 → 15 → 12	3
7	34	34 → 27 → 15	4
8	15	15 → 34 → 27	4
9	12	12 → 15 → 34	5
10	27	27 → 12 → 15	6
11	34	34 → 27 → 12	7
12	15	15 → 34 → 27	8
13	15	15 → 34 → 27	8
14	15	15 → 34 → 27	8
15	12	12 → 15 → 34	9
16	8	8 → 12 → 15	10

Using this table, we can easily answer the following questions:

A. How many misses does the cache experience for this reference stream? Show your working to obtain this answer.

Solution: 10 misses in total.

B. Which are the pages left in the cache after this reference stream? Ordering doesnt matter.

Solution: Page 8, 12 and 15 are left in the cache.

Question 2

Repeat Question 1 for the following reference pattern:

$$24, 25, 26, 27, 24, 25, 26, 27$$

A. How many misses will the cache experience? Show your working to obtain this answer.

Solution: The cache experiences 8 misses (i.e. everyone is a miss!):

Step	Current access	Cache (most recent $\rightarrow$ least recent)	Miss count
0	-	$\varnothing$	0
1	24	24	1
2	25	$25 \rightarrow 24$	2
3	26	$26 \rightarrow 25 \rightarrow 24$	3
4	27	$27 \rightarrow 26 \rightarrow 25$	4
5	24	$24 \rightarrow 27 \rightarrow 26$	5
6	25	$25 \rightarrow 24 \rightarrow 27$	6
7	26	$26 \rightarrow 25 \rightarrow 24$	7
8	27	$27 \rightarrow 26 \rightarrow 25$	8

B. How many misses would the optimal cache replacement policy (OPT) experience? Show your working to obtain this answer.

Solution: The optimal caching replacement policy discards the page that will not be referenced for the longest time in the future. This policy would experience 5 misses.

Step	Current access	Cache (after access)	Miss count
0	-	$\varnothing$	0
1	24	24	1
2	25	24, 25	2
3	26	24, 25, 26	3
4	27	24, 25, 27	4
5	24	24, 25, 27	4
6	25	24, 25, 27	4
7	26	26, 27, x	5
8	27	26, 27, x	5

where x is an unknown placeholder for whatever future references in the stream may be.

Question 3

You are the designer of a distributed system that does caches short video segments in their entirety from a server over the Internet. Video segments have an approximately normal size distribution, with a maximum observed size of 100 MB, a minimum observed size of 1 MB. The mean of the distribution is roughly 50 MB. Your server is located in Omaha, Nebraska which is roughly the center of the continental United States. Your clients are spread all over the continental United States (CONUS). Your deployment excludes Alaska and Hawaii. Akamai reports that the average end-to-end Internet bandwidth in the United States is about 15 Mbps in CONUS. About 95% of the accesses from users are for viewing a video, while 5% are for editing the videos after viewing it. Video playback typically consumes 0.5 MB of a video file per second. An edit session lasts at most 20 minutes, and the entire video segment is written back to the server at the end of the edit session.

Solution: We make the following assumptions:

- The propagation latency between server and clients can be neglected, because its very small comparing with transmission latency.
- 95% of operations are read and only 5% are write. So we make a design similar to read-write lock that favors read more than write, using lease.
- Clients cannot start viewing until they receive whole video segments.

A. Describe the design of a lease-based cache coherence protocol for this system that preserves one-copy semantics at the granularity of an entire video segment. You can

make simplifying assumptions, but your protocol must provide one-copy semantics at the stated granularity. Be sure to clearly state and justify any assumptions you make. Ideally, you would like to avoid lease renewal for videos that are only viewed (i.e. not edited). Explain how to achieve this goal.

Solution: A brief description of an acceptable protocol:

On server side, for each view lease request from client, if a edit lease is already granted, we put the view lease in the end of the queue, otherwise grant the view lease request.

On client side, if the view lease is granted by server, client can fetch the latest video from server for viewing. The view lease never expires until revoked by server. Otherwise, client plays the old version of video.

On server side, for each edit lease request, if theres a edit lease already granted, put the new edit lease request in the queue. Otherwise, server revokes all view leases, then approve the edit lease.

On client side, if the edit lease is granted, the client can fetch the latest video and make modifications. After editing is done, client sends the video back to server and terminates the lease.

Note: Students can also assume a streaming view of the video while downloading. This is acceptable, as long as the overall protocol ensures *one-copy semantic*.

B. Suppose lease requests that cannot be honored immediately are FIFO-queued at the server. What is the worst case waiting time for a lease request if the queue length is zero on arrival?

Solution: Worst case scenario:

1. If there is a brand-new edit lease given out by the server when the new lease request arrives, the server cannot give out any new leases until that edit lease expires such that the one-copy semantics are preserved.
2. The video segment associated with that brand-new edit lease is of size 100MB, which results in a relatively large transmission time.

Therefore, the worst case waiting time is:

$$\text{Edit time} + \underbrace{\text{Transmission time}}_{\text{round trip}} = 20\text{min} + \frac{2 \cdot 100\text{MB} \cdot 8\text{bits/byte}}{15\text{Mbps}} \approx 1306.67\text{s}$$

C. How does your answer to B change, if the queue length is 5 on arrival?

Solution: The worst case scenario:

1. There is a brand-new edit lease given out by the server.
2. The pending requests in the queue are all edit lease requests and the size of video segment associated with those edit leases are all 100MB.
3. The new lease request is a edit lease request.

So the worst case waiting time is (5 on the queue and 1 currently in process):

$$(5+1) \cdot (\text{Edit time} + \underbrace{\text{Transmission time}}_{\text{round trip}} = 1306.67\text{s} \cdot 6 = 7840\text{s}$$

D. What is the best case waiting time, if the queue length is 5 on arrival?

Solution: The best case scenario:

1. The current edit lease expires as soon as the new request arrives.
2. All the pending lease requests in the queue are view leases.
3. The new lease request is a view lease request.

So the best case waiting time is 0.

Question 4

Deciding to go retro like DropBox, you replace the caching mechanism described in Question 3 with full replication. Each client now needs to allocate sufficient local storage to contain all the video segments. Consider a point in time when there are one million video segments in the system.

A. Approximately how much storage will each client need to allocate for the video segments?

Solution: From Q3, the mean size of a video segment is 50 MB. So 1 million segments needs $50\text{MB} \cdot 1,000,000 = 50\text{TB}$ storage.

B. Suppose there are 10,000 continuously active users in the system who generate view and edit requests as stated in Question 3. Roughly how much update data will a client receive per second? You can make simplifying assumptions, but be sure to state them clearly.

Solution: From Q3, about 5% of users would edit the video. Assume there are $5\% \cdot 10,000 = 500$ users who are editing the video and each editing session takes 20 minutes. After which, the edited segment is immediately written back to server and then sent to all other users. And then, 5% of users start to edit again. So and so forth.

Therefore, in every 20 minutes, there will be $500 \cdot 50\text{MB} = 25\text{GB}$ updates, which translates into

$$\frac{25\text{GB}}{20 \cdot 60\text{s}} = 20.83\text{MB/s}$$

Note: assumptions can vary wildly, and so can the answer.

C. Suppose user behavior changes so that 50% of the requests are now for edits. Recompute your answer to B. under these circumstances.

Solution: Let all other assumptions still hold. Update rate will be

$$\frac{50\% \cdot 10,000 \cdot 50\text{MB}}{20 \cdot 60\text{s}} = 208.3\text{MB/s}$$

Note: basically the answer should be 10 times of the answer in B.

Question 5

Consider a server that hosts a read-only database. The database has 100,000 blocks, each 4KB in size. One of these blocks is a descriptor block; ten of the blocks are index blocks; the remaining blocks are data blocks. A client accesses this database using a cache of size of 1000 blocks, each 4KB in size. On a cache hit, accessing a block takes one ms. On a miss, it takes 100 ms. Suppose the client workload is as follows. Ten

percent of the accesses are to the descriptor block. Another ten percent are uniformly distributed over the ten index blocks. The remaining 80% of the accesses are uniformly distributed over the rest of the database blocks. In steady state, what is the average time for a client operation? (*Hint: you may need to make simplifying assumptions. Be sure to state them clearly.*)

Solution:

- *Assumption*: We assume the cache policy follows a consistent policy, say LRU, and the block access is approximately uniform in general within each category of the block. Also, we assume the accesses across categories are also uniform (so things such as 80,000 consecutive data block accesses followed by 10,000 consecutive descriptor block accesses would NOT happen). Thus, we can suppose an descriptor block access is detected for each 10 accesses, on average. The same reasoning applies to the other two blocks.

- *Analysis*: Since descriptor block and index block access is very frequent (1 in 10, and 1 in 100, respectively), we can assume that they are always in the cache (i.e. they are never the "least recently used"). Therefore, we can assume the rest of the 989 blocks are occupied by the data blocks, which are constantly brought to the cache and evicted later. Note that this is only

$$\frac{989}{100000 - 11} \approx 9.89 \cdot 10^{-3}$$

of all the data blocks. In other words, for a random block out of all the 99,989 data blocks, the expected time needed for accessing it is:

$$\begin{aligned}\mathbb{E}[\text{time for a random data block access}] &= 9.89 \cdot 10^{-3} \cdot 1 + (1 - 9.89 \cdot 10^{-3}) \cdot 100 \\ &\approx 99.021\text{ms}\end{aligned}$$

On average, there is a 80% chance we are accessing a data block, 10% chance for descriptor block or index block (both assumed to be in cache). Therefore:

$$\mathbb{E}[\text{time for any random block access}] = 99.021 \cdot 0.8 + 1 \cdot (0.1 + 0.1) \approx 79.4166\text{ms}$$

Note: Students may make different assumptions about cache policy and access distribution pattern.

Question 6

Give an example of a failure-free scenario in which the NFS v3 caching protocol (i.e., "faith-based caching") violates one-copy semantics. Your example should be detailed enough to clearly show step by step how the violation of one-copy semantics occurs.

Solution:

- *Assumption*: NFSv3 Faith-based caching has a time out of 3 seconds for regular file.
- *Scenario*: Consider the following case, where the `Time` entry shows a timeline:

```
Time     |= 0 ======== 1 ======== 2 ======== 3 ======== 4
Client A |= R ++++ W ++++++++++++++++++++++ CHK
Client B |= ---------- R ++++++++++++++++++++++++ R ++ CHK
```

- *Analysis*: In this case, one-copy semantics is violated, since ideally if client A and client B were operating on the same host, client B would be able to observe whatever changes made by client A, as client As write operation takes place prior to client Bs read operation.
- *Alternative solution*: Write-Write conflict: Two clients writes to their own cached copy.

  However in the scenario above, since NFSv3 clients cache data block for a certain amount of time, it is possible for W-R sequence to cause inconsistency between cached versions.

Note: Student must explicitly state that the operation takes place on the same data block since NFSv3 caches on block granularity rather than on file granularity. The key idea to this problem is that there are unforeseen writes in the process.