CS5412 Final Exam (May 4 version).

75m, with generous extra time for those who need longer or have SDS accommodations.

Closed book, no use of Internet. But you can use one page of home-made nodes. We want your answers to be as short as possible. Long essays usually include a mix of things that are correct and things that are wrong or unrelated, and we deduct for those EVEN if you had all the correct information somewhere in the answer.

Q1. (20 pts). The motor vehicle databases for a set of states are being merged for the first time. You start by uploading the data into HDFS and then use Apache Hive to create a single sharded HBASE table that has one row for every violation. This very large table is sharded into **row groups** and **column groups**. You are designing a job to list drivers who have a total of more than 4 violations in any single 12-month period between 2015 and 2022.

a. **[5 points]** In one sentence each, what is HDFS? What is HBASE? How does HBASE relate to HDFS?

HDFS is the Apache file system, part of the infrastructure that supports Hadoop.

HBASE is a technology for storing huge tables in HDFS.

HBASE actually maps tables to HDFS, by sharding them into "tablets" with some number of columns and some number of rows each, and then saving the tablets as files. For example, a big database might get sharded into 5 groups of rows and 5 groups of columns, yielding 25 tablets – each stored as an HDFS file.

b. **[5 points]** What does it mean to say that HBASE is sharded, given that HDFS itself was already sharded? What is a column group? What is a row group?

A tablet is a c x r table containing c columns from the original big table (column group) and r rows (row group). The HDFS table is literally cut into rectangular chunks (shards).

c. **[5 points]** A MapReduce/Hadoop/Spark job can only access one chunk of data at a time in the map step, so for this table, map can only run on a single column group and row group at a time. But this means that infractions for a single driver might show up in multiple different chunks. How would you use MapReduce to collect the data for each individual driver?

My map function would just extract (licence#, infraction) records. During the shuffle exchange, all the infractions for the same licence-number will be grouped, and my reduce function could then output a list in which a license# is included only if the particular driver exceeded the threshold.

d. **[5 points] Without writing any code**, just in words, tell us what LINQ is and how you could now design a LINQ expression to solve the problem.

Given a licence-# and a list of infractions, sort the list by date, then use a where clause to form a LINQ collection of infractions for a 12-month ranges where the range starts from some infraction and goes to that date in the next year. Count all infractions in this range. If this is ever greater than 4, add the licence# to our list of drivers flagged for too many violations. Now we can do further processing, or just print this set by iterating over the servers where we computed it.

Q2. (20 points). We learned that key-value systems such as CosmosDB and Cascade often have a versioned tuple replace in which you call **put** with some version number, and the tuple will be stored only if the version matches what you specified.

True or false (+5 points for a correct selection, -2.5 for an incorrect selection)

a. ***TRUE.*** Using versioned **put** a set of tasks can update a shared (key,value) object such that every update will be performed exactly once, and will be totally ordered.
This was the technique we learned about early in the semester and used in HW2.
b. ***FALSE.*** Compared to locking, the average delay for a versioned replace will be thousands of times less.
Thousands? That is a very big number… But it definitely could be smaller.
c. ***TRUE.*** Applications written to use a versioned **put** need a loop.
Definitely. If a put fails, the application generally has to try again.
d. ***TRUE.*** Compared to locking, the versioned replace has simpler behavior if an application crashes while updating an object.
With locking we would have to worry about how to deal with a partial update or a lock that will never be released because the lock holder crashed.

Q3. (20 points). FLP studies the problem of fault-tolerant consensus on a single bit, 0 or 1, with halting (crash) failures. In an asynchronous network that never loses a messages, the result shows that a correct protocol cannot be proved to terminate if even a single process might (or might not) crash.

a. **[5 points]** Tell us what "correct" means in this situation.

A correct consensus protocol is one where all non-crashed participants deduce the same decision value, and at least some participants voted for that value.

b. **[5 points]** Tell us what termination means in this situation.

A consensus protocol terminates if all the non-crashed participants eventually deduce a decision value.

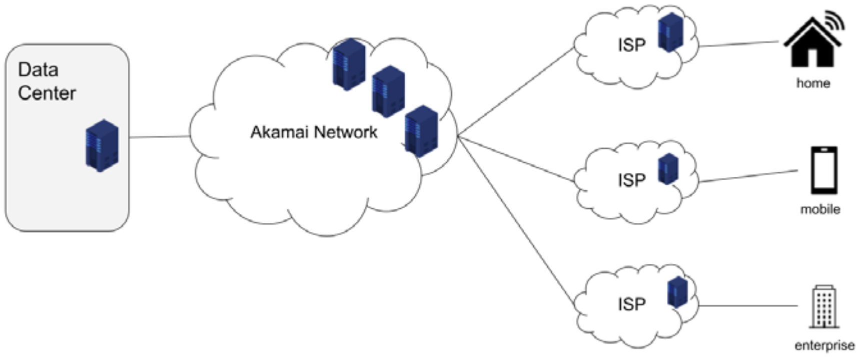
c. **[5 points]** Focusing on Paxos, we know that a write only has to reach a majority of the logs (the "acceptors"), and a read only needs to read enough logs to overlap with the writes. Does FLP apply to a program that uses Paxos as follows: the *n* processes all try and write their vote into the first Paxos log slot, and whichever one wins, we pick that value as the decision.

FLP does apply. Our leaders could contend, indefinitely, to write to the same slot. We didn't ask you to tell us how the FLP attack might look, but it would cause ballot numbers to increase without limit. However, a perfect adversary would be needed.

d. **[5 points]** Same scenario as part c. A criticism of FLP is that the attack it describes isn't very likely. If we take this into consideration, is it reasonable to say that Paxos is live after all?

Paxos still cannot <u>guarantee</u> progress if a partition breaks the service into parts such that no server can contact a quorum. So Paxos is still not live in a formal sense. But we probably wouldn't worry about it!

Q4 (20 points). [Akamai] We had a guest lecture with Anna Blasiak, an architect from Akamai, who shared details about load balancing in the Akamai content delivery network (CDN). As a reminder, this was Anna's slide showing the architectural structure of Akamai's CDN:



On the left is the data center of Akamai's customer (also called the "origin datacenter" in some of our lectures. Akamai maintains its own network, seen in the middle, while also making agreements with Internet Service Providers (ISPs) who host Akamai CDN machines directly in the ISP's datacenters. The end-users are on the right.

a. **[5 points]**Give two examples of the data that can be stored at an ISP's site.

Images and videos. Advertisements.

b. **[5 points]**Explain how this design can benefit the end-users who access Akamai-based applications.

Cached content can be retrieved quickly from a nearby Akamai site, so this speed up the user experience.

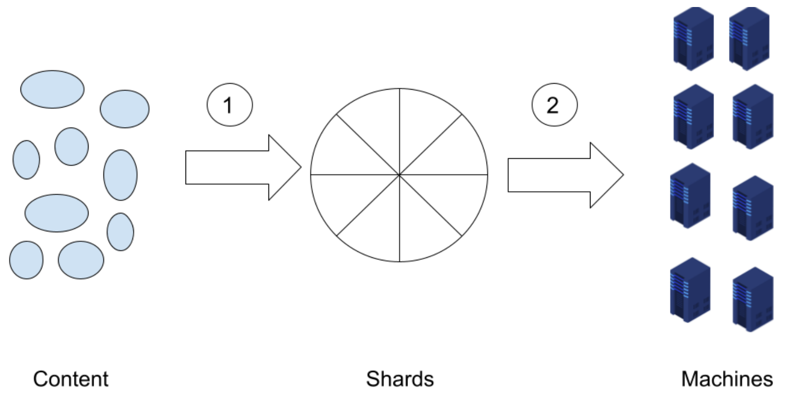
c. **[5 points]** Does this design have any benefits for Akamai or to the ISPs, as they operate this structure? Explain.

Akamai saves money by not needing to create its own datacenters close to the end-user, and is able to offer better quality of service because the ISP point of presence will be physically close to the user. The ISP benefits because the content delivered by Akamai won't have to travel through their entire network, which reduces overall network load and lets them focus their bandwidth on other kinds of data.

d. **[5 points]**What issues might arise with this ISP hosting approach that Akamai would not experience if it ran the entire structure itself, and didn't place Akamai machines in ISP data centers?

Akamai won't be able to control many aspects of the runtime environment where its edge machine are physically placed, and if something breaks, like a network link, might not be able to service it rapidly (it might need help from the ISP staff). If Akamai had its own infrastructure, it could fix anything.

Q5. (20 points) Now let's take a closer look at the cloud datacenter. Inside the datacenter, Akamai utilizes sharding for their distributed data storage system. However, unlike the key-value examples we saw in other lectures, Akamai thinks of its sharding as occuring on a circular space of hashed keys, as seen in the illustration below.



In this graphic, the Akamai-hosted content is on the left. In step ① Akamai hashes the key for each content item, placing the key-value pair along the ring at the numerical location that the hashed value takes us to (the numbers wrap at the top). The ring is split into shards – our graphic shows eight shards. Step ② is managed by the team Anna works on, and involves storing the DHT into the servers, which are shown on the right. Select the statements that are correct.

True or False (+2 for each correct selection, -1 for each incorrect selection)

a. **True.** Because Akamai uses random hashing (like SHA 256) for step ①, and the number of key-value objects is enormous (billions), the mapping of content to shards should be relatively even, with roughly the same number of objects per shard. For example we would not see double the number of objects in one shard compared to the other shards.

   We could certainly see small variations, but if the number of objects is so large, each shard will have lots of them, and you would not expect huge variation.

b. **False.** Random hashing has a second benefit, which is that the expected request rate (load) for each shard should be quite balanced too. For example, we would not expect that a shard is somehow getting twice as many access requests as some other shard.

   This is not at all correct. Think about a situation in which some photo is very popular. Everyone wants to see it – that specific key is getting accessed a LOT.

c. **False.** Akamai places the servers along the ring in a single round-robin manner: machine 0 handles shard 0, machine 1 handles shard 1, etc. (Note: C and D cannot both be true or both be false).

   Akamai uses an algorithm Anna's group "owns" to decide which servers will hande which shards and could assign one shard to multiple servers, or one server could even handle multiple shards.

d. **True.** Akamai places the servers along the ring in a company-developed manner that involves running an algorithm that assigns each server to some "spot" on the ring. (Note: C and D cannot both be true)

   Same point as in part c.

e. **True.** When an object is placed into the ring, the *k* servers along the ring in clockwise order will host it and handle requests to it. *k* can vary for different objects based on how popular they are.

Definitely. For example if Akamai wants two replicas of each object, it just makes a copy on the server closest to where that object key hashes, and then on the next server in clockwise order.

f. **True.** Because of E, a new server will need to receive a *state transfer* from some other server to initialize it. The state would be the set of objects it needs to have copies of, to host them.

   Right. This is how we avoid having the new server be "cold" which would cause a crazy burst in cache misses.

g. **True.** Suppose that a server becomes unusually loaded. In this situation, Akamai can change where the servers reside in the mapping described in d, enabling them to give more load to a lightly loaded server or to spread load from an overloaded one.

   Totally valid. This is what Anna's team does.

h. **True.** If an image or video exists in multiple sizes, there will be a different cached object for each size of the object and Akamai will treat them independently.

   This is how they do it.

i. **False.** As an additional caching feature, Akamai leverages caching on nearby mobile phones or laptops: if an application requests content that isn't available in Akamai, it will be fetched directly from a peer device first (one of the other systems on the extreme right in the first graphic). This ensures that content will only be requested from the originating data center if Akamai has never seen it before.

   Akamai never does this kind of client-to-client direct sharing. It would violate trust and privacy and also run down the batteries of a mobile host very rapidly.

j. **True.** When a company uses Akamai CDN hosting, it still needs to be prepared for situations in which Akamai is temporarily not soaking up much (or any) load, so it might see some bursts of requests to its origin servers from time to time.

   True, and Facebook turns out to have seen this (it came up during her lecture).

Q6. Extra credit, up to 5 points, counts only towards points lost on exams. Why did Microsoft create a new WiFi protocol, what was their key idea, and how did the resulting "white"-Fi protocol get used in FarmBeats?

Microsoft discovered that on a farm, normal WiFi routers just don't have enough reach to connect to sensors or tractors out in the field. By repurposing unused television spectrum as WiFi channels (white noise channels, hence White-Fi), they gained more WiFi spectrum, including some low frequency channels that have really long range – providing networking access for the entire farm. Farmbeats uses this to connect to drones and tractors that are out in the field, away from the barn.