Name: _____ Netid: _____

Quiz 1

Open book / notes, but you must not ask anyone else for help.
Released at noon, 3-18-21. Due by noon 3-21-21.

This should take 75-90 minutes, but we are giving you 2 hours to reduce the sense of time pressure. Download the answer sheet from CMS, edit it, and then upload the completed version. We use logs to verify that you uploaded it within 120 minutes after downloading. If you have a SDS accommodation, your time will be extended accordingly.

Type your answers, and keep them short and clear – we aren't looking for essays.

Q1 problem setting: You are in charge of content caching for an internet news feed. Your system runs on the Azure cloud and implements a DHT cache, with one node per shard (no replication). If a failure causes some node in the DHT to crash, a replacement is automatically launched and will jump into the empty slot. It will start up with an empty ("cold") cache. *If we scale this form of cache up by adding more nodes, we would have more shards but they would still have one node per shard.*

The way your service works is very simple: any time the newspaper has a new article, web page servers check the DHT first for a copy of any photos or video in the article. If they get a hit in DHT cache, they don't need to fetch it from the newspaper database. On the other hand, if the DHT cache lacks the photo, the server will fetch the photo from the database, but then stores a copy into the DHT. This way other servers won't need to pester the database.

[1] As the newspaper becomes more and more popular, you increase the number of DHT server nodes in proportion to the peak number of concurrent customers. Would you expect this to provide linear scalability? Explain, and be sure to state any assumptions that your answer depends on.

[2] Prince Harry and Duchess Meghan have a new baby, and they release an official photo. Your DHT service malfunctions: instead of showing the official photo, the news article shows a blank box with an X in it and a message "DHT service timeout error." You notice that one of the DHT server nodes has crashed. Worse, each time it self-repairs, the new server crashes too. Explain what is causing this problem, and why your answer to part [1] makes sense given this obvious scaling issue!

[3] Suppose that you are permitted to modify the implementation of the DHT cache server, but cannot modify the application (the newspaper web implementation, or the photo collection service). How could you change or reconfigure it to fix the caching issue identified in [2]?

[4] Now, unlike for part [3], suppose that you *cannot modify the DHT cache in any way*. How could you modify the newspaper photo management service (and if needed, the first-tier web servers that generate web pages) to avoid the issue identified in [2]?

[5] Exactly at the same time that the DHT cache was having problems due to the Harry and Meagan photo, lots of other photos were displaying without problems. On the other hand, there are some

photos that had problems too. For example, a photo of Professor Birman teaching cloud computing that happened to be in the Cornell alumni news ("Professor still teaching after nearly 40 years") also times out, just like the Harry and Meagan one. Explain both observations.

Q2: DHT hashing. In class we looked at key-value stores (DHTs) that map from key to shard this way: **shard-number = Hash(key) % number-of-shards**. Here, **Hash** is a hashing function supplied by the programming language, implemented by a method such as SHA-64 or SHA-256.

[1] Define the term "shard" that was just used above.

[2] A shard often uses state-machine replication for fault-tolerance. What guarantees do state machine replication methods provide?

[3] Suppose that an application needs to store a photo into the DHT. The developer proposes to hash the photo itself (the jpg image object) and use the hash as a key. Would this be a good idea? Why?

[4] Suppose that we insert 75,000 objects into the DHT, each with its own unique key. If there are S servers (assume S < 20) and each shard has just one server (so there are also S shards), would we expect each shard to have exactly 75000/S objects on it? Would it point to a bug if some shard had twice as many objects on it as the average shard, and some other shard had half as many? Explain.

[5] Suppose we have sub-objects, like "Ken Birman/photo", "Ken Birman/age", "Ken Birman/job". These names are used as keys. Would you expect to find all of the sub-objects on the same shard, or spread over multiple shards? Explain briefly.

Q3: Jim Gray concern. In a paper we discussed, Jim Gray and some colleagues at Microsoft studied a model of a database with T concurrent transactions and S servers, and discovered that in their model, overheads were growing as a polynomial (specifically, S^3T^5). You just took a new job at "Skunkworks.com", and your team is responsible for the main database system. Nobody at your new job is aware of the Jim Gray paper or analysis, so the team has never seen this formula.

[1] Your boss says that the system currently has a capacity of T = 10,000 transactions per second and runs on a server with S=2 nodes. Load is growing and by the end of the year is predicted to reach 25,000 transactions per second. The team wants to upgrade to a 5-node system (S=5), and your boss asks you to double check that the new configuration will be more than fast enough for T = 25,000 transactions. Is Jim Gray's analysis useful here? Why or why not? Will the upgrade help? *Keep in mind that your boss and the team had never heard of Jim's result. They could be doing a smart thing, or they could be doing a dumb thing… they have never seen Jim's equation.*

[2] Your boss believes that the current load limit of T = 10,000 is actually due to "hot spots" in which many transactions happen to access the identical item at the identical time. The team has come up with an idea for eliminating half the hot spots (they can't eliminate all of them, so transactions will still be accessing some shared data). Will this hot-spot-reducing idea increase the capacity of the server? Explain your reason for saying yes or no.

[3] Your team has decided to take Jim Gray's advice and shard the database. Explain what this actually means. For example, suppose that the original single database had 10M database items in it, and suppose that the sharded system will have 5 shards. Where do the 10M items end up stored?

[4] Your boss asks you to port some of the existing transactions to the new sharded system. You notice that several transactions to atomic transactional (SQL) reads or updates to data that would now live on multiple shards. What issues will arise when you port this logic to the sharded solution? *Hint: your answer should talk about performance issues but also correctness, for example if some of those transactions do reads and updates to multiple data items.*

Q4: Paxos Protocol. Leslie Lamport's original Paxos protocol defines two constants, QW and QR.

[1] Define the constants QW and QR, including any restrictions they must satisfy.

[2] In very short English sentences, one sentence each, define the Paxos "client", "leader", "acceptor" and "learner" roles.

[3] What possible values for QW and QR are permitted in a Paxos system with 5 acceptors that *needs to remain available even with* 1 failed node (1 acceptor might not be accessible)? N*ote: reads and writes must remain active and complete successfully even if only 4 of the 5 acceptors can respond.*

[4] Now consider a sharded service like a DHT and assume that it uses Paxos inside the shards, to implement state machine replication. Suppose that the shard size is 3, but that the service as a whole has 300 members (so, it has 100 shards). In your answer to [1] you talked about the number of members. Would QW and QR need to be defined relative to 300, or relative to 3? Explain.

[5] Suppose that Paxos was deployed once per shard in our system for [4], but also deployed one extra time for the whole service (spanning the full set of shards). The idea is to use state machine replication for the shards, but also to have configuration data that the whole application uses, and the "whole service" Paxos would be used to track changes to the configuration data.

a. Would the whole-service Paxos use the same log as the per-shard Paxos, so that a given slot in some shard's log could be either a shard-data update or a configuration update? Explain.
b. When using Paxos this way, would the total-order guarantee of Paxos span all updates (shards and also the configuration updates), or would configuration updates not be ordered relative to shard data updates?

Q5: IoT Data and the Meta System. Suppose we have IoT sensors that have some known error bound for their value, δ, and also some known error bound on time, ε. Thus if a sensor reports value x at time t, this really needs to be interpreted as $x \pm \delta$ at time $t \pm \varepsilon$.

[1] Suppose that we have redundancy: some value is tracked by 3 sensors. 1 might be faulty. Meta teaches us that the correct sensor values will overlap. Thus if 1 sensor gives a very divergent reading, we can conclude that the other 2 must be correct. Explain why this might let us get a more accurate estimate of x and t than the official accuracy of $x \pm \delta$ at time $t \pm \varepsilon$.

[2] Still in a case where we might have 1 faulty sensor out of 3, suppose that all 3 sensor readings overlap (meaning: any pair of 2 has at least some shared value and time range). What can we conclude about the actual ("true") value of the underlying data?

[3] Suppose that we need to cool a chemical reaction if the temperature *might* have reached some threshold value *max*. In case [2] where we have 3 sensors *and they all overlap*, what rule should we use to be sure that we never make a mistake and let the chemical reaction overheat? *Hint: before answering, make a picture of 3 sensor readings that all overlap, and think about what it means if you suspect that perhaps 1 of the 3 is in fact a faulty sensor. Keep in mind that the batch of vaccine will be spoiled if the reactor temperature actually does exceed max. We don't want the faulty sensor to be able to trick us into allowing that to happen!*

[4] In a dairy setting, suppose that we measure the body temperature of a cow using a sensor inside the cow's stomach (this doesn't bother the cow). Now assume that the cow drinks a gallon of cold water and the sensor temporarily shows the water temperature, until it warms up. Thus in this case the sensor is not reporting what we would like it to report – the value is a "faulty" value for the cow's body temperature. Would the 2 out of 3 redundancy feature allow us to figure out the actual cow body temperature (e.g. if we put 3 sensors in the cow's stomach instead of just 1)? Explain.