SENG 468 Assignment 2

Marcus Dunn

1. Instructions

1.1. Running

Run the application with docker-compose. All the containers are hosted on github container regestry publicly.

```
docker compose up
```

1.2. Testing

The files to run the commands are located in the `scripts` directory. They can be run in any order - it won't break things - but to see each feature in action it is reccomended to run them in the order they are numbered. The responses from the server will be printed to console.

2. Questions

1. What are the trade-offs between using a normalized schema versus a denormalized schema in MongoDB? Which approach would you recommend for the social media platform database and why? (5 points)(O4)

   A normalized schema allows for better data consistancy as there is a single source of truth for each data point. It makes updates much simplier as there is only one place to update.

   A denormalized schema has the benifit of being easier to evolve over time. There is also no need for joins which can improve preformance (especially in a distributed setting).

   For a social media platform I would reccomend (and went for) a mostly denormalized schema. As there is few consiquences to having inconsistant data in a social media site the ease of schema evolution and preformance gains are worth the pains involved.

2. How would you design an index in MongoDB to support a query that searches for all posts with a particular tag? How would this index be impacted if the number of posts in the database grows significantly? (5 points)(O4)

   Each post has an array of string tags. I would create an non-unique multi-key index on the array. The documentation is here and suggests that the lookup preformance would stay $O(\log(N))$ in the number of index entries.

3. Suppose you need to add a new field to the posts collection to track the location where the post was created. How would you modify the existing documents in the collection to include this new field? What are some potential issues that could arise from this modification? (5 points)(O6)

   I would not edit documents without the information. The application would have to handle the case where a location is not specified. I imagine regulatory bodies would mandate that this would remain an option even after the update.

   This adds additional logic in the applicatio but is much preferable to adding misleading and incorrect data to the database (such as the classic of setting coords to 0,0 by default).

4. How would you use Redis as a cache for frequently accessed data in the social media platform? What are the benefits and drawbacks of this approach? How would you handle cache invalidation and cache expiration? (5 points)(O6)

I set the cache to expire after an hour, the caches are populated on reads and updates. There could be issues with post edits (which are not currently supported!) but provided the application remembers to update the cache on edits it would be fine. An hour provides a sufficent amount of time for high-loads to be handled gracefully (such as a popular user posting a new post) by guarenting cache hits for the first hour (and as the cache is repopulated on new reads a long-running traffic to a single post will still be mostly cache hits.

This stragegy finds a balence of invalidating somewhat agressivly to keep cache size down (and remove the chances of having to hit the database often due to redis having to remove cache entries for popular posts) while still keeping things around in cache for enough time for it to be useful.

5. How would you use Kafka or RabbitMQ to handle real-time notifications and messaging between users on the social media platform? What are the benefits and drawbacks of each messaging system? How would you ensure message persistence and replication? (5 points)(O6) I used RabbitMQ. This was chosen because it is a lot simplier to use and the library support is better in rust. If I had a much larger system that could not afford to keep queues nearly empty (as is optimal with rabbit) or I needed the disk persistance that kafka defaults to I would choose kafka instead.

   I ensured that the messages were persisted by acknowledging the message once it was persisted to the database.

6. In a multi-user environment, how would you handle concurrency control and data consistency between MongoDB and Redis in the social media platform? What are the benefits and drawbacks of this approach? (5 points)(O4)

   There was very little protection against inconstantacies between the cache and mongo. As the cache invalidates itself I considered a request successful (and returned a response as such) if the database transaction succeeded. This allowed the application to contunue functioning in the face of cache failure. The cache invalidation by time guarnteeded that the database changes would eventually be syncronized with the cache and the up to a hour of inconstancy was determined to be acceptable given the application domain.