Assignment 3: Report

Riyaadh Bukhsh 921470997

Teammates

Riyaadh Bukhsh

Statement about ChatGPT

I used ChatGPT to help with insight into syntax of the json file structure, primarily with the collection `listings_with_cal`. Also, I discussed with GPT on ideas to optimize the performance of the indexing as well as retrieval of documents. This helped enormously when retrieving fro the database, as we know the effects of indexing. Please refer to the references and the notebooks files provided by the TA and the discussions were used.

Statement about distributed work

I did all the work for this assignment. I am a one-man team.

Comments on Observed Performance

As an individual team, I will write a short paragraph for the following questions based on my work on this assignment.

1. For Part 3, if you include the index then the pipeline runs in 1 or 2 minutes, but if you leave the index out then it would take about 4 hours. For this question, assume that on a particular machine it takes 2 minutes with index, and 4 hours without index.

    - Compute the (approximate) time it takes for MongoDB to make one full scan of the `db.reviews` collection.

      Answer: My approximation for a full scan of db reviews would be around 4 hours since each document would be scanned.

    - Compute the (approximate) time it takes, on average, for MongoDB to perform an index-based retrieval of all documents in `db.reviews` having a particular listing id value.

      Answer: Index-based retrieval would be about 2 minutes since the index based scanning is exponentially faster.

2. For Part 2 we did not use an index. Why does your pipeline for Part 2 run in a minute or two, even though the `calendar.csv` file has many more entries than the `review.csv` file?

   Answer: This is most likely because of the pipeline aggregation. This aggregation processes an enormous load of data which can simplify the data retrieval process.

3. For Part 4 we again did not use an index. Why does your pipeline for Part 4 run in a minute or two, even though both collections being joined have 39K+ documents in them?

   Answer: Similar to the scenario for Part 2,the collection has more than 39K documents each, if the pipeline still takes about a minute or two to run without the use of indexes, it would have to do with the nature of the join or the operations that are being carried out. For instance, if the join is based on a simple match that the database system can optimize out, or there is partitioning or possibly an efficient scan that the database system employs.

4. Would your pipeline for Part 4 run faster if you included an index on id for one or both of the collections?

   Answer: I have experimented with this concept and, yes of course indexing on id for both collection speeds up the processing time ten-fold. Indeed indexing helps a ton.

5. In Step 5, item 1(b) the question is asked: "Why do you think that creating the result (a cursor) is so quick but creating list(result) takes so only?". What is your answer to this question?

   Answer: It is more efficient to create a cursor since it simply prepares a reference to the result set on the server, but retrieving all results with list(result) will not happen quickly due to the necessity of retrieval, parse and storage of every bit of information on the client side.

6. In Step 5 you did an informal comparison of the running times of Query 5 pos and Query 5 neg, which do not use any index, and Query 7 pos and Query 7 neg, which takes advantage of the text index. (It was an informal comparison, because you ran the queries only once, rather than a lot of times, which would reduce the influence of random factors on the observed run times.) Assuming that your informal results are more-or-less representative of a more rigorous benchmarking experiment, explain the results obtained. How does this compare with the similar experiments performed in Programming Assignment 2 involving the tsv-based text indexing?

   Answer: Overall, the results are way more efficient in processing time, with and without indexing. This is attributed to the nature of NoSQL databases. Now if we are comparing relative to normal relational databases (which I would deem an effective comparison) then here is my analysis.

Query 5 pos/neg (without index): These queries perform full collection scans.This makes them slow and computationally expensive.
Query 7 pos/neg (with text index): These queries use a text index whijc in turn allows for fast lookups by leveraging precomputed mappings of words to documents.

References

1. GeeksforGeeks. *How to Create Index for MongoDB Collection Using Python*. Retrieved from `https://www.geeksforgeeks.org/how-to-create-index-for-\\mongodb-collec`

2. MongoDB, Inc. *MongoDB Cheat Sheet*. Retrieved from `https://www.mongodb.com/developer/products/mongodb/cheat-sheet/`