TDT4225 – Exercise 1

Vladislav Levitin

1. **How does the Flash Translation Level (FTL) work in SSDs?**

   The FTL is responsible for mapping logical block addresses (LBAs) to physical block addresses (PBAs). When a host requests data from an SSD, the FTL translates the LBA to corresponding PBA, and theSSD reads data from that location. The FTL is typically implemented in the firmware and is specific to the type of SSD model. The FTL is one of key components that makes the SSD work.

2. **Why are sequential writes important for performance on SSDs?**

   Sequential writes are important for the performance on SSD, because they allow the SSD to write data in large, contiguous blocks. This maximizes the number of I/O operations per second (IOPS) that the SSD can perform and reduces the amount of time required to write data to the SSD.

3. **Discuss the effect of alignment of blocks to SSD pages?**

   Alignment of blocks to SSD pages can have a significant effect on the performance. If the blocks aren't aligned, the SSD has to perform extra work to read and write the data. This can cause a significant decrease in the performance.

4. **Describe the layout of MemTable and SSTable of RocksDB.**

   RocksDB uses a Log-Structured Merge-Tree (LSM-Tree) to store data. The data is first written to a log-structured memtable and then flushed to an SSTable. The memtable is a sorted key-value store and the SSTable is an immutable file that is sorted by key. MemTable is a sorted or unsorted string map. It is stored in a format that allows RocksDB to perform prefix compression on keys. SSTable is a sorted string table. It is stored in a format that allows RocksDB to perform compression on both keys and values.

5. **What happens during compaction in RocksDB?**

   Compaction in RocksDB is the process of merging and rewriting data in a database to reduce the amount of space it takes up. This is done by combining data from multiple files into a single file, and then rewriting that file to be more efficient. During compaction, RocksDB will first create a new memtable and write all of the data from the old memtable to the new memtable. Once the new memtable is full, RocksDB will flush the contents of the memtable to disk and then remove the old memtable.

6. **Give some reasons for why LSM-trees are regarded as more efficient than B+-trees for large volumes of inserts.**

   LSM-trees are generally faster for inserts than B+-trees because they require fewer disk operations. This is because LSM-trees write new data to memory first, and only flush to disk when the memory buffer is full. B+-trees, on the other hand, must write new data to disk immediately. LSM-trees also tend to be more space-efficient than B+-trees. This is because LSM-trees only store the data that has changed in memory, whereas B+-trees must rewrite the entire block of data every time a change is made. Finally, LSM-trees can be more easily parallelized than B+-trees. This is because LSM-trees can flush data to disk in parallel, whereas B+-trees must flush data sequentially.

7. **Regarding fault tolerance, give a description of what hardware, software and human errors may be?**

   Hardware errors can include things like a faulty power supply, a bad memory module, or a malfunctioning hard drive. Software errors can include things like a buggy driver or a virus. Human errors can include things like accidentally deleting a file or misconfiguring the system.

8. **Give an overview of tasks/techniques you may take/do to achieve fault tolerance.**

   Fault tolerance is the ability of a system to maintain its functionality when one or more of its components fail. There are many ways to achieve fault tolerance, including redundancy, voting, and checkpointing.

9. **Compare SQL and the document model. Give advantages and disadvantages of each approach. Give an example which shows the problem with many-to-many relationships in the document model, e.g., how would you model that a paper has many sections and words, and additionally it has many authors, and that each author with name and address has written many papers?**

   The SQL approach would be to have three separate tables: one for papers, one for sections, and one for words. Each table would have a foreign key column linking it to the other two tables. This approach would be advantageous because it would be easy to query the data and to update it if necessary. However, it would be disadvantageous because it could be difficult to model the relationships between the data. For example, it could be difficult to model the relationship between a paper and its authors.

The document model would be to have a single document that contains all of the data for the papers, sections, and words. This approach would be advantageous because it would be easy to model the relationships between the data. However, it would be disadvantageous because it would be difficult to query specific sections of the data and to update it if necessary. For example, it would be difficult to update the data for a paper if the author's name or address changed.

10. **When should you use a graph model instead of a document model? Explain why. Give an example of a typical problem that would benefit from using a graph model.**

    A graph model should be used instead of the SQL model when data is interconnected and there are complex relationships between data points. A typical problem that would benefit from using a graph model is a social network, where data points represent individuals and relationships between them represent the different ways they are connected.

11. **You have the following values for a column: 43 43 43 87 87 63 63 32 33 33 33 33 89 89 89 33**
    a. **Create a bitmap for the values.**

       32: 0 0 0 0 0 0 0 1 0 0 0 0 0 0

       33: 0 0 0 0 0 0 0 0 1 1 1 1 0 0

       43: 1 1 1 0 0 0 0 0 0 0 0 0 0 0

       63: 0 0 0 0 0 1 1 0 0 0 0 0 0 0

       87: 0 0 0 1 1 0 0 0 0 0 0 0 0 0

       89: 0 0 0 0 0 0 0 0 0 0 0 0 1 1

    b. **Create a runlength encoding for the values**

       32: 7, 1

       33: 8, 4

       43: 0, 3

63: 5, 2

87: 3, 2

89: 12, 2

12. **We have different binary formats / techniques for sending data across the network:**
    a. **MessagePack**
    b. **Apache Thrift**
    c. **Protocol Buffers**
    d. **Avro**

    In case we need to do schema evolution, e.g., we add a new attribute to a Person structure: Labour union, which is a String. How is this supported by the different systems? How is forward and backward compatibility supported?

    a. MessagePack: Supports schema evolution by allowing fields to be omitted when serializing and deserializing. This allows for both forward and backward compatibility.
    b. Apache Thrift: Supports schema evolution by allowing fields to be added, removed, or renamed. This allows for both forward and backward compatibility.
    c. Protocol Buffers: Supports schema evolution by allowing fields to be added, removed, or renamed. This allows for both forward and backward compatibility.
    d. Avro: Supports schema evolution by allowing fields to be added, removed, or renamed. This allows for both forward and backward compatibility.