TDT4225 – Solution to Exercise 1

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

   The FTL in SSDs handles the mapping between logical disk addresses and physical disk addresses. The idea is to do what is called “wear leveling”, e.g. ensure that writes are spread around the disk to wear each part of the disk equally. This is done by doing garbage collection, e.g. blocks containing many invalid pages, are copied to freshly erased blocks, creating blocks with many free pages. The original block is then erased and finally made available to new allocation. Thus, at high write activity, the SSD will have to a lot of garbage collection to ensure that enough blocks are available for writes of pages.

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

   This has to do with garabage collection, because it becomes more efficient in terms of larger areas. With sequential writes, the data in the pages of the block becomes invalid at the same time. As a result, data usually does not need relocation during garbage collection since there is no valid data remaining in the block before it is erased. It also has to do with the alignment of writes to blocks. If the writes can easily be aligned to blocks, there will be no need for extensive copying of pages during garbage collection.

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

   The size of the write requests is important to consider as it has an impact on performance. A write request should ideally be a multiple of the clustered page size, Write requests that are aligned to clustered pages can be written to disk with no further write overhead. In contrast, write requests that are not aligned, generate overhead because the SSD controller needs to read the rest of the content in the last clustered page and combine it with the updated data before all the data can be written to a new clustered page. Such read-modify-write operations increase write latency.

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

   MemTable and SSTables have the same layout, thus, MemTables may written directly as they are.

Figure 2.19: The block-based table format in RocksDB.

They consist of N data blocks and some administration data. The most important is an index to the sorted data in the data blocks. This is organized as a skip list, being similar to a B+-tree. In addition, a bloom filter may be used to easily filter out most requests to keys which are not present in this MemTable/SSTable. Additionally, compression of data will/may be used.

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

   Compaction is important in RocksDB. During compaction data in one level of the LSM tree is moved down to the level below. This is done to let the most recent data be available in the highest level of the LSM-tree. In RocksDB there is a multi-threaded compaction going on, meaning that there are multiple threads doing compaction of different parts of the tree. Duplicate keys are removed during compaction, so that the most recent of them is kept.

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

   This is mainly due to the fact that larger chunks of data is written out in LSM trees. While updates in the B+-tree may spread updates all over the tree, updates in the LSM tree are localized to the MemTable, which in turn will be written as a complete chunk to disk. However, during compaction some extra work will happen in the LSM tree.

7. Regarding fault tolerance, give a description of what hardware, software and human errors may be?
   Hardware errors may be anything related to hardware, like disk failures, memory failures, network card failures, etc.
   Software errors are problems with software not doing what is expected. This could be things happening to unusual input, due to long running times, bad programming and testing practice, etc.
   Human errors happen typically when the operation of a system is going wrong. May be the interface is not intuitive, may be the user is not completely awake, etc.

8. **Compare SQL and the document model**. Give advantages and disadvantages of each approach. Give an example …..

   The document model like used in MongoDB, lets web data be easily modelled and stored together as one chunk in the database. This is advantageous when a document is created and used together as a chunk. The locality of the data will be good, and this creates the possibility for efficient reads and writes of this type of data. The disadvantage of the document model is that inter-document relationships, e.g. many-to-many relationships, are hard to create. Here, the SQL model is better, where you may join almost anything in the database. Thus, by using foreign references, relationships to any data may be created. The query model where you use a high level language which is compiled into an efficient query execution plan, is a good way of creating

an efficient access to almost anything in the database. The main problem with the SQL model is that you have to use a pre-defined schema, such that when data layout changes, you have to convert data in the database.

Authors write Papers containing paragraphs and words. Here, it would be obvious to model the paper as a document and Authors as another document. To connect these two concepts, we would either use a reference mechanism from Authors to Papers, and possibly vice versa. These may be joined in the application. Another possibility is to denormalize (e.g. authors are copied into papers), but this creates problems of update anomalities, as you know from normalization theory.

9. When should you use a graph model instead of a document model? Explain why.

   A graph model is advantageous when the data is a graph, e.g, data is highly related to other data through links. When the queries are of the type traversing links in the graph, graph databases are better than using the document model. In Kleppmann several examples are given, like the social graph, web graphs, road or rail networks, etc. Vector drawings would also benefit from a graph model.

10. **Column compression**: 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 0000000100000000
    33 0000000011110001
    43 1110000000000000
    63 0000011000000000
    87 0001100000000000
    89 0000000000001110

    b) Create a runlength encoding for the values

    32 7,1
    33 8,4,3,1
    43 0,3
    63 5,2
    87 3,2
    89, 12,3

10. We have different binary formats / techniques for sending data across the network:

    - MessagePack

- Apache Thrift
- Protocol Buffers
- 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?

MessagePack: This not good for schema evolution since it does not have a schema specification. It is simply binary encoded JSON data.

Thrift and Protocol buffers: Both support schemas and use a code generation tool to create classes in various programming languages. Both use tag numbers for the different fields. If a tag number is missing when decoding a message, it may be ignored. When new tag numbers are used for schema changes, both forward and backward compatibility will be supported, but fields may be omitted at decode time. For backward compatibility, every field that you add must be optional or provide a default value. You may only remove fields which are optional.

Avro: Supports schemas through Avro IDL. Forward compatibility means that you have a new version of the schema as a writer and an old version of the schema as a reader. Backward compatibility means that you have a new version of the schema as a reader and an old version of the schema as a writer.