Very Large, Distributed Data Volumes (TDT4225)

Course Coordinator : Svein Erik Bratsberg

Assignment 1

Submitted by
Name : Md Anwarul Hasan
Student ID : 583233

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

Answer:

Solid state drive or SSD is a storage device and unlike the traditional hard disk drive (HDD) it does not have any moving component in it. This SSD come with a firmware which is known as flash translation level (FTL). This FTL implements wear leveling and wear leveling is responsible for writing in the whole disk instead of repeatedly doing erase-write cycle over the same set of blocks.

This wear levelling are three types. Those are given below,

i) No wear leveling: Here the mapping from operating system's logical memory to physical memory is fixed. To make modification of data, there are several steps, i.e. block need to be fetched, then erase on the disk, after that that make change in the memory and finally write back in the original location of the disk.

ii) Dynamic wear leveling: In dynamic wear leveling, mapping from logical to physical address is done dynamically. When updating a block, the original block is marked as invalid and and updated block is written in new location.

iii) Static wear leveling: It performs like dynamic wear leveling but also moves tatic blocks periodically.

FTL also runs garbage collector to make pages available for writing later. This garbage collector erases a block when all pages of a block are invalid. But as lot of block may contain both valid and invalid pages, the garbage collector algorithm decides whether certain percent of a block is invalid and then writes the existing valid pages to a new block and finally erase the old block.

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

Answer:

Because during sequential writing on SSD performance does not degrade and lifetime increases.

Bothe the writing technique, i.e. sequential & random, performs almost same at the beginning. But, the after a certain time the garbage collector start making latency in random write technique.

As the garbage collection factor is not an issue of sequential write and sequential write prolongs disk's lifetime that is why sequential write is important for performance of SSDs.

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

Answer:

The alignment of write request has significant important on performance of SSD. If the write request is multiple of clustered pages then the request will be written in the disk without any overheads and so there is no latency.

On the other hand, if the write request is smaller or larger than the clustered page size, then the SSD controller need to read the rest of the content in the clustered page and combine it with the updated data before writing all the data to a fresh clustered page. These read, then modify and finally write adds latency during write operation.

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

Answer:

MemTable and SSTable are described below.

SSTable: The original concept of LSM tree is implemented in RocksDB and are called SSTable. The default SSTable is block based table that stores sorted key-value pairs with metadata and indexes. At the beginning of the file there is data file which stores key-value pairs. The next block is meta block that stores Bloom filters, statistics, compression and other metadata. Next is meta index block that contains one entry for each data block. An in the index block there is a key which is a string and is greater than or equal to the last key in the data block and before the successive data block that enables a search for correct data block within the index.

MemTable: The original concept of LSM-tree has a component C_0 and when LSM-tree is implemented in RocksDB, the C_0 is called the memtable. Memtable accepts new updates until it becomes full and when it is full RocksDB creates new memtable and mark old memtable as frozen, which means it is immutable. All the entries of a frozen memtable are written to disk as SSTable immediately or later based on the configuration. The default memtable index is a skip list that stores its entries in sorted order, so that it can be searched by binary search and will also be effective while writing in the disk in SSTable.

5. RocksDB: What happens during compaction in RocksDB?

Answer:

Compaction is equivalent process of rolling merge process of LSM-tree. It takes two or more SSTable as input, merge their entries to get the complete ordering that is defined in an comparator and finally write the joined entries in new immutable SSTable. In the output deleted marked keys are removed. However, if a key persist in two SSTable then only the latest value is included in the output.

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.

Answer:

There are several advantages of LSM-trees over B+ trees for large volume of inserts. Some of them are given below,.

1st, the B-tree index writes all the data twice, i.e. at the write-ahead log and another is the tree itself. Moreover, B+ tree updated the entire page even if few bytes of that page is changed. This also adds overheads.

2nd, LSM-trees have lower write amplifications than B+ trees, so LSM-trees can perform better write throughput. The other reason of LSM-trees better write throughput is that is sequentially writes compact SSTable where as B+trees rewrite several pages.

3rd, LSM-trees are not page oriented and periodically rewrites SSTable to avoid the fragmentation. Thus is it utilizes the disk space more efficiently, because B+-trees leaves some disk-space unused when page slits or when a row can not fit into a existing page. As a result LSM trees can be compressed better also.

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

Answer:

Hardware errors could be hard disk crash, faulty RAM, blackout of power grid, faulty network devices etc.

Software errors could be a bug in the software itself, that can cause crash for all the instances of a application serve for a bad input. The other forms of software failure are problematic runway process that shares resources, faulty service of a system and cascading type failure where one small failure causes failure in another component and this triggering continues.

Humans are regarded as non-reliable entries. They are prone to mis-configuration which is the pioneer cause of problems.

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

Answer:

Achieving 100% fault tolerance is not possible in reality. However we may take few steps to make the system more fault tolerant. Some of the techniques or tasks are given below.

The avoid hardware error, we could apply redundancy in each and every steps so that if one fails another keep the process running. In the disk we may apply RAID so that failure in one disk may not hamper data on the other disk. Data-centres may have generators and uninterrupted power supply backup batteries. Servers may also have dual power supply and hot & swappable CPUs.

Software error are more critical while thinking fault tolerance. End to end testing, carefully thinking about interactions & assumptions, isolation of processes, testing the system for crashes for test cases, monitoring & measuring the performance of software in production may help achieve fault tolerance for systems.

To minimize human error several steps can be taken. One of them is designing system in a way that human get less change to do error. Letting people practice in a safe place where they can not cause harm the production environment but can explore and experiment safely may lessen the human error. Thoroughly testing in all phases, enabling quick recovery from human errors, setup proper monitoring like system performance and error rates, and good management with proper training may also lessen the human error.

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?

Answer:

Document model of database stores data in documents. Here database schema is not imposed. This document database is faster than SQL database while accessing data. It is field-based and has easy replication support. However, Complex joins like many-to-many is not good in document model.

Relational model or SQL stores data in rows. It requires pre-determined schema. Though it is slower than document model but it performs better in many-to-many relation situation. Relational model is column based and it does not have fast replication support.

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.

Answer:

I would use a graph model instead of document model when the data in my application shows many-to-many relationships. Because document model efficiently can handle tree-structured data, i.e. one-to-many, or when there is no relation between data or simple many-to-many relation. The support for joints is poor in document database.

The graph model has vertices or nodes and edges or relations. So by using this feature we can easily represent social graph where vertices will represent people and edges will represent to whom the people are connected with.

11. 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.

b) Create a runlength encoding for the values

Answer:

a) The bitmap is created below,

Column values																
Product_sk	43	43	43	87	87	63	63	32	33	33	33	33	89	89	89	33
Bitmap for each possible value																
Product_sk=32	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Product_sk=33	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	1
Product_sk=43	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Product_sk=63	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
Product_sk=87	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Product_sk=89	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

b) Runlength encoding of the of the above ae given below,

Product_sk=32: 7, 1 (7 zeros, 1 one, rest zeros)

Product_sk=33: 8, 4, 3, 1 (8 zeros, 4 ones, 3 zeros, 1 one)

Product_sk=43: 0, 3 (0 zeros, 3 ones, rest zero)

Product_sk=63: 4, 2 (4 zeros, 2 ones, rest zeros)

Product_sk=87: 3, 2 (3 zeros, 2 ones, rest zeros)

Product_sk=89: 12, 3 (12 zeros, 3 ones, rest zeros)

12. 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?

Answer:

MessagePack : It is a binary encoding for data interchange. It is fast and small in compare to other encodings. Binary encoding is 66 bytes long.

Apache Thrift and Protocol Buffers: both are libraries of binary encoding. Both of them need schema for data to encode.

Avro: Is also a binary encoding database. But not similar to apache thrift and protocol buffers. Avro has two different schema language, i.e. one for human editing and another for machine readability.

Schema evalution in apache thrift and protocol buffers: While doing schema evaluation, apache thrift and protocol buffers accepts new field as long as there is a new tag in every new field and it is not mentioned as required. The old code skips the data written by the new code, even if there is a new tag, while reading. This feature supports forward compatibility. As this new field has new tag numbers the new code can always read old data. And to support backward compatibility every new field must have a default value of should be marked optional.

Schema evaluation in Avro: here the forward compatibility means the new schema will act a writer & old schema as reader and vice verse for backward compatibility. In avro, to ensure compatibility, only fields with default value can be added or removed otherwise if we add a value that has no default value, new readers will fail to read data written by old writers thus it will break backword compatibility. On the other hand, if we delete a value that has no default value, then old readers will not be able to read data written by the new writers thus hampering forward compatibility.