Delta lake

Cloud has been successfully able to entice multiple commercial vendors due to its easiness such as scaling and decoupling storage and compute and due to its cost effectiveness. Cloud object stores such as S3 and azure blob storage are very popular and are used to store huge data warehouses and data lakes. But this can be problematic if we need support for ACID transaction with strong Consistency guarantees and listing objects. Eg. Parquet file has footer containing meta data information helpful for range but due to latency of key value stores, this metadata operation can be slower than actual queries. This is due to the fact that the popular object stores have their implementation as key-value stores making this features difficult. Key value stores have no cross key consistency typically used in encryption systems. Delta lake is an open source project, by Databricks tries to solve this issue by providing an ACID table storage layer. It provides this feature with the help of a transaction log(write ahead log) which is itself stored in the cloud object storage in the apache Parquet format. Due to the open source parquet file format, the delta lake tables can be accessed from major big data systems using connectors such as Hive, presto and spark and also supports important features such as time travel, data layout optimization, upserts etc. Since the object in the Delta table and logs are immutable, this feature can be exploited for caching at local storage.
Fig 1. successfully shows the contrasts achieved by using the “lakehouse” paradigm.

Key-value store mostly support eventual consistency model for each key and no consistency across keys. Even if strong consistency is provided, atomic operations are still not supported. Different cloud provider provides different form of consistency. One example given is of S3 where it supports read-after-write consistency. But even that carries an exception due to its negative caching. Snowflakes tackles this by separately managing metadata in a strong consistent service. But this approach can have multiple issues such as maintaining such external “high” performant service, more engineering work to maintain connectors. Hive has similar approach using the Hive Metastore in ORC format. Delta lake stores transaction log and metadata in the cloud object storage to achieve serializability.

A delta lake table is basically a list of files and a log containing transactional operations. It supports checkpointing to provide backup and uses optimistic concurrency control protocol. The data object are in apache parquet format, the log is in the _delta_log subdirectory and log checkpointing for performance. The log is compressed and the compressed version is called checkpointing. Delta lake supports serialization and all transaction that supports write are serializable due to the choice of concurrency control protocol. Delta lakes has various advance features mentioned in section 4 especially the time feature and data layout

optimization such as z ordering and OPTIMIZE. It offers various connectors to Spark SQL and Sparks Data source API.The manifest files similar to Apache Hive exposes a static file to expose the static snapshot. It has huge amount of use cases discussed in the section 5 and has impressive benchmarks. It has certain limitations such as currently only provides serialized transaction within single table. Delta lake is also dependent on external services and that can be issue for streaming data. Also it does not support secondary indexes yet.

Effectively, Delta Lakes provide an ACID transaction layer to the cost-effective cloud object stores often saving significant amount of money and engineering time of its customer

Comments:
The time travel feature sounds very similar to the lineage in RDD. Here they are storing the checkpoint in a table like structure. Shouldn't storing it like a graph be more effective to track?
Also the caching is also very similar to Sparks for adhoc queries by loading the data in-memory at local nodes.
I am kind of confused why spark is mentioned here. Spark is effectively made for corse-grained workloads. Delta lake is for transactional workload which is fine grained. Why would this two be used in the same context? I am not sure if I am missing something and would appreciate some clarity here.