CptS 415 Big Data
Assignment 4
Jinyang Ruan
011696096

1. NoSQL
   a. Explain the concept of noSQL
      NoSQL refers to a non-relational database. Unlike traditional relational databases, NoSQL does not guarantee the ACID properties. NoSQL is also sometimes understood as " Not Only SQL".
   b. Describe the 4 types of noSQL systems; for each category, give an example noSQL system (try to give different system from what we introduced in the class).
      - **Key Value:** Key-value database uses a hash table with a specific key and a pointer to a specific data (value). For example, GitHub (Riak), Instagram (Redis).
      - **Column Store:** Column storage database stores data in a column family, and a column family stores relevant data that is often queried together. Column families are usually stored in a big table. For example, eBay (Cassandra), Facebook (HBase).
      - **Document-oriented:** Document-oriented databases store data as documents. Each document is a collection of data items. Each data item has a name and a corresponding value. A document database can be viewed as an updated version of a key-value database, allowing nested key values, and having more efficient queries. For example, Codecademy (MongoDB), NBC News (RavenDB).
      - **Graph (beyond noSQL):** Graph database allows data to be stored as a graph. Entities will be used as vertices, and relationships between entities will be used as edges. For example, T-Mobile (Neo4J).
   c. Pick one type of noSQL system you give in b, give a real-world application that can make best use of the system, and an application that may not be well-suited for the system. Explain your reasons.
      I would like to choose **column storage databases** to discuss. A good application which can make best use of the system is ***Twitter***, it uses HBase and Cassandra to store user information and tweets. Their advantage is that: Column families can store tweets' tags, categories, links, and comments in different columns. This helps to speed up queries. In Twitter, strong scalability is necessary because of the mass amount of data. In addition, the storage of data with the same properties in a single column provides the possibility for large-scale data analysis.
      On the other side, Cassandra's data structure is based on how we expect the data to be queried. It is not possible to determine how the query pattern will change at the beginning of development. Once the query pattern changes, the design of the column family changes. This will affect the project development schedule. In addition, a system that requires write and read operations in an "ACID transaction" is not appropriate. If the system such as ***airline platform system*** frequently needs to read every record row by row, the column storage database is not a good choice.

2. KVstore
   a. Describe the challenges of designing a KV store.
      - **Fault tolerance→**replication
        Handle machine failures without losing data and without degradation in performance.
      - **Scalability→**serve get()'s in parallel; replicate/cache hot tuples
        - Need to scale to thousands of machines
        - Need to allow easy addition of new machines
      - **Consistency→**quorum consensus to improve put/get performance
        Maintain data consistency in face of node failures and message losses.
        Consistency is not trivial to achieve in distributed systems
      - **Heterogeneity** (if deployed as peer-to-peer systems)
   b. What are the two basic operations in a KV store?
      - $put(key, value)$
      - $value = get(key)$

3. Column Store
   a. Explain the major features of column stores in terms of data storage and storage key.

      **Data storage:** Column store data are stored in the disk column by column. In a column-store, all values of one column are stored sequentially on a database page. I/O efficient for read-only queries as they read, only those attributes which are accessed by a query. Most of the queries does not process all the attributes of a particular relation.

      **Storage key:** To answer queries, projections are joined using Storage keys and join indexes. Within a segment, every data value of every column is associated with a unique Skey. Values from different columns with matching Skey belong to the same logical row.
      - The column store does not generate redundant data during the reading process.
      - During the query process, concurrent execution of column store can reduce the query response time. And the ability to find data efficiently in data columns without the need to maintain indexes.
      - Minimizing irrelevant I/O during the query to avoid full table scanning.
      - Since each column is stored independently and the data type is known, compression algorithm can be dynamically selected for the data type and data volume size of the column to improve the physical storage utilization.
   b. We introduced three techniques to optimize column-oriented databases: compression, late materialization, and block iteration. Please explain how they work.
      - Compression: Because the data types are the same, the column store can be compressed. Lossless compression can be implemented according to the dictionary document. And content with low entropy can have a higher compression ratio.
      - Late Materialization: This technique is mainly to solve the problem of how to achieve maximum data filtering and reduce unnecessary IO and memory consumption without indexes.

- Block Iteration: Change traversal tuple to traverse the entire block of data. For a fixed-size column, you can think of it as a tuple. Minimizing the overhead of tuples by taking advantage of the ability to store parallelism in columns