CPT_S 415

Big Data, Fall 2020

Homework # 4

Tayyab Munir

11716089

1. **This set of questions get you to engage fundamentals of noSQL systems.**
   a. **Explain the concept of noSQL.**

      NoSQL is a non-relational database management system, different from traditional RDBMS. The NoSQL term should be used as in the Not-OnlySQL and not as No to SQL or Never SQL.
      [1] Main advantages of NoSQL are the following aspects:
      1) reading and writing data quickly.
      2) supporting mass storage.
      3) easy to expand.
      4) low cost.

      Also, NoSQL databases are classified in-between the spectrum from ACID to BASE. ACID and BASE are discussed later in this assignment.

   b. **Describe the 4 types of noSQ systems; for each category, give an example noSQL system (try to give different system from what we introduced in the class).**

The four types of noSQL systems are as follow:

1. Key-value store: Key value store noSQL database is the type of noSQL database that uses hash table in which a unique key exists and each unique key point to a particular item of data. The storing of the keys is done in a bucket and there can be identical keys in different buckets. It is helpful to use but this model or type of database does not provide any traditional database capabilities and if the data volume increases it will be difficult to maintain the unique key values. Both keys and values be can be anything, simple objects to complex compound objects. Redis, Amazon S3 (dynamo) are the most well-known system in this category.

2. Document Store: Document store noSQL database is the type of noSQL database that stores documents, and these documents is a collection of key value pairs compressed as a document. IT is quite like key value store, but the only difference is that the values stored in the document provides some structure and encoding of the managed data, XML, JSON, BSON are used for encoding. Couchbase and MongoDB are the most popular document-based databases.

3. Column-based store: Column based store noSQL database is the type of noSQL database that store data together as columns instead of rows and are optimized for queries over large datasets. The most popular are Cassandra and HBase.

4. Graph-based: Graph based noSQL database is the type of noSQL database that are used to store information about networks such as social communications. Neo4J is a good example of this database.

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.

Document store:

A real-world example where document store can be used to store heavy data such as digital content, articles, contents, metadata and ebooks. In these cases, the applications data is the most heavily accessed and the data needs to be accessed in less response time. Using document store database for these kind of applications gives the flexibility to both store the data in XML, or JSON and provide faster access to the data.

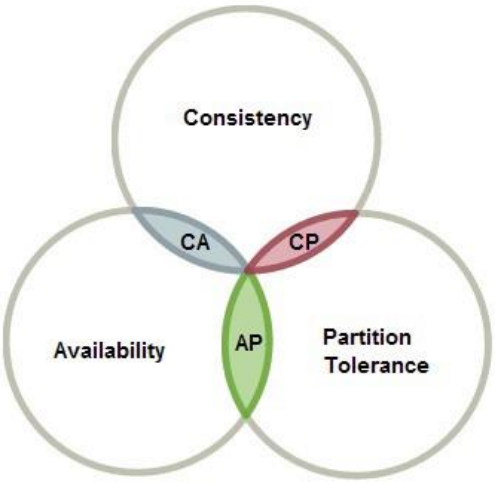
In banking area, we cannot use noSQL database we need data accuracy, funds transfer, data integrity, multi-record state and many other features where noSQL is a good fit, rather we can use traditional database.

2. **This set of questions are related to data consistency.**
    a. **What is CAP Theory? Consider a toy example of a cluster that contains two servers S1 and S2. Use the example cluster to explain the CAP theory. You can simply use the proof I explained in class.**

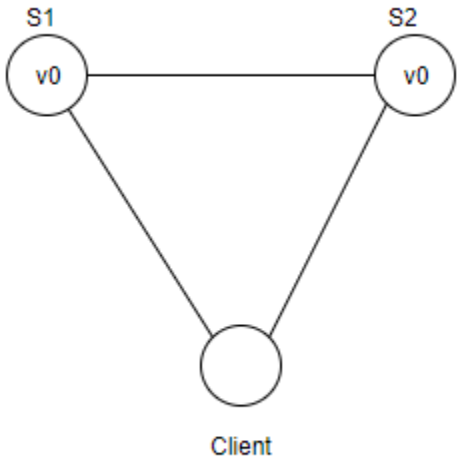
CAP theory:

[4]The CAP theory is a tool used to makes system designers aware of the trade-offs while designing networked shared-data systems. CAP has influenced the design of many distributed data systems. It made designers aware of a wide range of tradeoffs to consider while designing distributed data systems. The theorem states that networked shared-data systems can only guarantee/strongly support two of the following three properties:

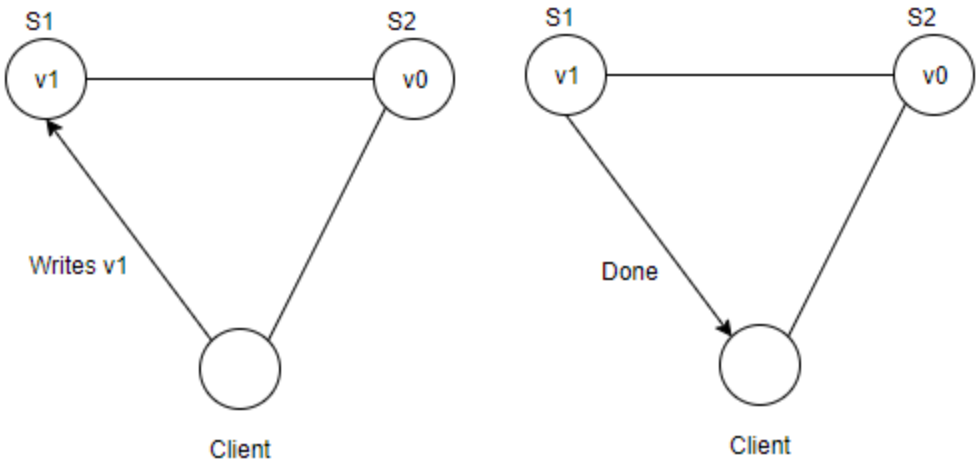
- Consistency - A guarantee that every node in a distributed cluster returns the same, most recent, successful write. Consistency refers to every user having the same view of the data.
- Availability - Every non-failing node returns a response for all read and write requests in a reasonable amount of time.
- Partition Tolerant - The system continues to function and upholds its consistency guarantees despite network partitions.



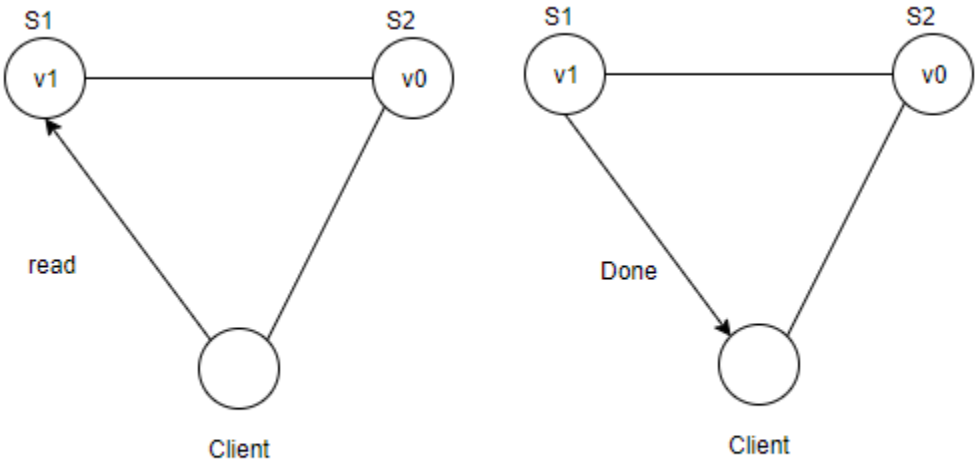
Let us consider a very simple example of distributed system. Our system is composed of two servers, S1 and S2. Both servers are keeping track of a toy V. The initial value of the server is v0. S1 and S2 can communicate with each other and can also communicate with external clients. Here is what our system looks like.



A client can request to write and read from any server. When a server receives a request, it performs any computations it wants and then responds to the client. For example:



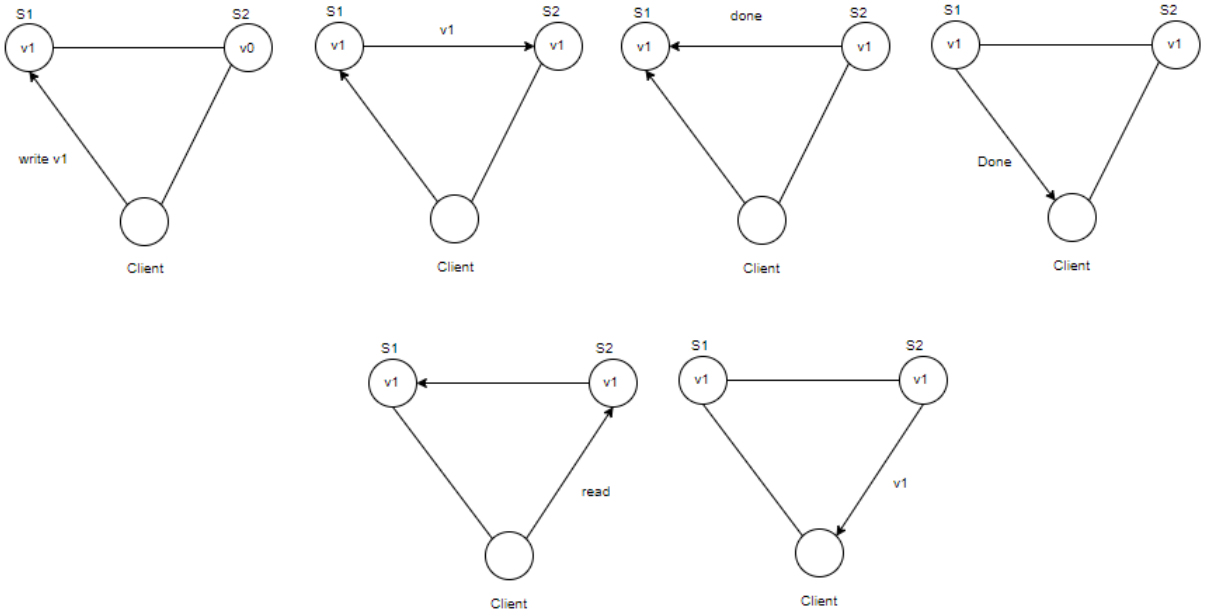
And the read operations look like as follow:



Now we are all set to explain the CAP theory through this example.

Consistency:

In a consistent system, once a client writes a value to any server and gets a response, it expects to get that value back from any server it reads from. For example:



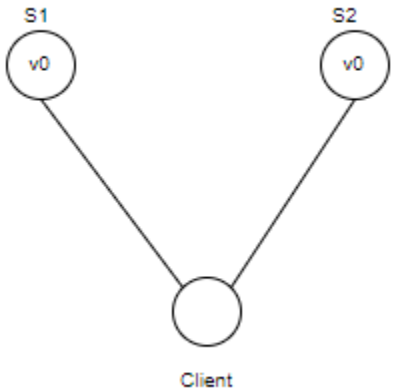
In this system, S1 replicates its value to S2 before sending an acknowledgement to the client. Thus, when the client reads from S2, it gets the most updated value of V: v1.

Availability:

In an available system, if client sends a request to a server and the server has not crashed, then the server must eventually respond to the client. The server is not allowed to ignore the client's requests.

Partition tolerance:

If any messages S1 and S2 send to one another can be dropped. If all the messages were being dropped, then our system would look like this.



This prove shows that a system can only have two at a time it can not have all three features.

b. Consider Consider the relation Accounts (acctNo, balance), and two SQL statements that conduct a request "transfer $200 from account 123 to 456". In a DBMS, this usually includes two steps:

i. Add $200 to account 456:
UPDATE Accounts SET balance=balance+200 WHERE acctNo = 456

ii. Subtract $200 from account 123:
UPDATE Accounts SET balance=balance-200 WHERE acctNo=123

Use this example and necessary scenarios to show when Atomicity, Consistency, Isolation and Durability can be violated.

ACID is a database design principle related to transaction management. It is an acronym that stands for:

- atomicity
- consistency
- isolation
- durability

ACID intends to guarantee validity even in the event of errors, power failures, etc. It ensures that consistency is maintained throughout the application.

Atomicity makes sure that all the operations always succeed or fail completely and have no partial transactions. The following steps shows that when Atomicity can be violated.

1. Users add $200 to account 456 before the occurrence of error.
2. An error occurred before subtracting the transferred amount from account 123 the database roll backs and partially effects the transaction and return the system back to its previous state.

Consistency guarantees that the database will dependably stay in a predictable state by guaranteeing that just information that fits in with the limitations of the database schema can be composed to the database. Therefore, a database that is in a steady state will stay remain in a consistent state resulting in a successful transaction. The following steps shows that when consistency can be violated.

1. The user tries to update the account with data type double which is of float value.
2. The database applies the validation check and as this violates the constraint checks it stops the transaction.

Isolation ensures that the results of a transaction are not visible to other operations until it is completed. The following steps shows that when Isolation can be violated.

1. User attempts to update two records one for acctno = 456 and the other one for 123.

2. The database successfully updated the first record.
3. However, before the systems update the second record another user changed the same record with some other value, so the second user's transaction will stop until the transaction performed by first user fails or succeeds.

Durability make sure that the output of an operation is permanent, such that once a transaction has been performed it cannot be rolled back. The following steps shows that when Durability can be violated.

1. A user tries to update the account 456 and add $200 to it.
2. The database validates the value and update it.
3. After the successful completion of the transaction, the systems return the updated record upon the request.

3. [Column-store] We take a closer look at Column store.

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

1. Data storage: In the column-store the data is stored in sequence of columns instead of rows. In a column-store, all values of one column are stored sequentially on a database page. Less data must be fetched from memory and Data compression might work better.
2. Storage Key: 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.

b. We introduced three techniques to optimize column-oriented databases: compression, late materialization, and block iteration. Please explain how they work.

4. [newSQL] (20)

a. Describe the definition of NewSQL systems and design principles.

New SQL databases are modern databases that solve some of the major problems associated with traditional online transaction processing (OLTP)RDBMS. They seek to achieve the scalability and improved performance of NoSQL databases while maintaining the benefits of traditional database management systems. NoSQL combines OLTP, high performance, and scalability of NoSQL.[2]

Design Principles of NoSQL are:

1. minimizing or stay away from locking
2. rely on main memory
3. try to avoid latching
4. cheaper solutions for High Availability

c. **For in-memory DBMS, describe the set-associative cache for block placement and LRU block replacement policy.**

With set-associative placement, there are more than one block to choose from on a miss.

Least-Recently Used (LRU) - to reduce a chance of throwing out information that will be needed soon, accesses to blocks are recorded. The block replaced is the one that has been unused for the longest time.[3]

Advantage: takes locality into account

Disadvantage: as the number of blocks to keep track of increases, LRU becomes more expensive (harder to implement, slower and often just approximated).

References:

[1] Jing Han, Haihong E, Guan Le, & Jian Du. (2011). *Survey on NoSQL database. 2011 6th International Conference on Pervasive Computing and Applications.*

[2] https://www.predictiveanalyticstoday.com/newsql-databases

[3] https://web.cs.iastate.edu/~prabhu/Tutorial/CACHE/bl_replace.html

[4] https://dzone.com/articles/understanding-the-cap-theorem