KTH Royal Institute of Technology
Stockholm

School of Electrical Engineering and
Computer Science

Data-Intensive Computing - ID2221

Review Questions 1

Author
Emil Ståhl

Author
Selam Fsha

September 12, 2021

Review Questions 1 - ID2221

Emil Ståhl and Selemawit Fsha Nguse

September 2021

1 1. Explain how a file region can be left in an undefined state in GFS?

A file region is consistent if all clients will always see the same data, regardless of which replicas they read from. The state of a file region depends on the success of an operation and whether there are concurrent mutations or not. Undefined state can occur due to failure or concurrent writes. Below is an in-depth description of each scenario:

Failure:

If an append operation fails at any replica server, the GFS client will retry the operation causing the replicas holding the same chunk to contain different data due to duplicates of a record. This can occur without having concurrent writes, simply as a result of a failed write which can occur due to a time out. The failed write that caused a duplicate record can also cause an undefined region. The undefined file region state can occur if a replica fails to recognize a mutation of a chunk, leading to that the replica does not perform the operation. When the client later retries to perform the append operation, the client that failed the last operation will have to pad the missing data in order for the new data / record to be written at the correct offset. This causes the replica to have padding written in a region where other replicas have the previously written data. Inconsistency causes an undefined file state.

Concurrent successes

As described above, a failed write can cause a file region to be inconsistent, hence undefined. But successful concurrent writes can cause consistent but undefined file regions. This can occur if the GFS client writes a very large chunk since it then breaks the chunk down into multiple write operations. This can lead to that the write operations are being interleaved or overwritten by concurrent operations from other GFS clients. This means that a file region can contain fragments from other clients, however, all replicas containing the chunk will hold

identical data due to that the primary node enforces one update order on all replicas. This results in a consistent but undefined file region state.

Failure is a fundamental problem of distributed systems. When a failure occurs the sender may not know if the recipient received the message or not. This is why distributed systems guarantee that a message is delivered at least once or at most once. In this case, the developers of GFS decided upon at least once delivery to the storage nodes which led to undefined and inconsistency are possible states for a file region[1].

2 Briefly explain the read operation in GFS? What's the role of the primary node?

Read Operation:

- Application initiates the read request. Application sends the read request to the GFS client with the file name the application wants to read.
- The GFS client translates the request to GFS syntax and sends it to the master node with the filename and chunk index.
- The master responds with chunk handles and replica locations. The master sends a handler for the chunk that the GFS client needs and the replica server locations where the chunks can be found.
- The client picks the most optimal replica server location for retrieving the chunks and sends a request for retrieving the desired chunk. The GFS sends sends the request directly to the chosen replica server with the chunk handler.
- The chunkserver or servers sends requested data to the client.
- The client forwards the data to the application
- The application can use that data.

Role of primary node:

When an application requests to modify a chunk, there needs to be a so-called primary node that defines the update order. The master node finds the set of chunk servers that have the desired chunk and grants a so-called **chunk lease** to one of these chunk servers. The chunk server that holds the chunk lease is defined as the primary node, the other chunk servers are defined as secondaries. There can only be one primary for each chunk. The role of the primary node is to determine the serialization (update) order for all of the chunk's modifications, and the secondaries must follow that order to ensure correctness. It is the job

[1]The Google File System Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, Google

of the primary to communicate the order to the secondaries. The primary node holds the chunk lease for a set period of time, approximately 60 seconds. After expiration, the master may grant the lease to another node or renew it. Master is also able to revoke leases to disable modifications when a file is being renamed. If the master loses contact with the primary node, it will grant the lease to another node once the lease expires. In short, the role of the primary node is to enforce one update order for all chunk servers holding a specific chunk[2].

3 Using one example, show that in the CAP theorem, if we have Consistency and Partition Tolerance, we cannot provide Availability at the same time.

The CAP theorem is an important theorem in distributed systems and databases. It states that it is impossible for a distributed system to have more than two out of the three following properties:

- **C**onsistency - Every read receives the most recent write or an error
- **A**vailability - Every request receives non-error response
- **P**artition tolerance - The systems continues to operate despite failures

This means that if we have consistency and partition tolerance in our system, we cannot ensure availability. However, it should be noted that in the absence of a network failure, when the distributed system is running normally, both availability and consistency can be satisfied. The theorem is often misunderstood as one has to choose two properties at all times, when in fact the choice between consistency and availability is only in case of a network failure. As an example we can take a hypothetical financial service, in the case of a network failure we can choose to fulfil either consistency or availability. Due to the nature of the industry when we try to read transactions or account balance information we want to have the latest write or return an error response. When the system results in error response or timeout errors(i.e. ensure consistency), it fails to meet the availability property. An example of the CAP theorem in the industry is when using BigTable, in case of a network failure it will only ensure consistency and not availability. Cassandra, on the other hand, prioritizes availability over consistency in case of a network failure[3].

[2]The Google File System Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, Google

[3]Perspectives on the CAP Theorem, Seth Gilbert, Nancy Lynch, Massachusetts Institute of Technology

4 Explain how the consistent hashing works.

Cassandra uses data partitioning when the size of an object exceeds the capacity of a single machine. To partition, Cassandra uses a technique called consistent hashing which determines on which node in the network the partitioned data should be placed, it works as follows:

The data and the node id are hashed using the same hash function which operates in the same ID space. The ID space is equivalent to the output range of our hash function and is treated as a fixed ring of indexes. Each node in the system is assigned an arbitrary id within the id space which represents its location in the ring. Each item stored in Cassandra identified by a key is assigned to a node in the cluster by hashing the data and then walking clockwise around the ring until a node is found that has a value larger than the item's key obtained from the hashing algorithm. Meaning that each node in the cluster is responsible for the space on the ring between it and its predecessor on the ring. The partition on which we should put our data is given by the following hash function:

$partition = \text{hash}(d) \bmod n$ where d corresponds to our data and n is the size of our ID space.

Let's assume a network of 5 nodes with the ID's {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, when we hash the data "ID2221" we get the partition 69, on which node should we put the partition? In consistent hashing with data partitioning, we put the partition on the node having the next ID that is higher than the partition. In this case, our partition 69 should be put on the node with the id of 70, since that's the next node when moving in a clockwise orientation around the id space ring.

To find an item in Cassandra, the application specifies the key of the item it wish to obtain and Cassandra uses it to route requests. However, the implementation of Cassandra results in some problems, since each node is given a random value on the ring it leads to a non-uniform distribution of items which affects load balancing[4].

[4]Cassandra - A Decentralized Structured Storage System, Avinash Lakshman, Prashant Malik, Facebook

5 Explain the finding process of tablets in BigTable.

In BigTable, database tables are split into multiple tablets which are segments of the table split at certain row keys so that each tablet is a few hundred megabytes or a few gigabytes in size. The tablets can then be stored on a distributed system of servers. To find a given tablet, the following process is used:

BigTable uses a three-level hierarchy for keeping control of tablets. The first step that is performed is to read a file stored in Chubby, which is a distributed lock manager used with BigTable to ensure that there is only one active master. The file stored in Chubby holds the location of the root tablet, this tablet is responsible for storing the locations of the complete set of tablets existing in the system and is treated differently since it is never split to preserve the hierarchy. The root tablet stores this information in a special metadata table. There are different levels of metadata tables, the root tablet uses its metadata tablet to point to other metadata tablets. Here, in the second level of metadata tablets is where BigTable store the locations of every tablet present in the system. The location of a regular user tablet is stored under a single row in the metadata tablet. These tablets are put together to form a complete table in the database. Lastly, to minimize requests and latency, the client using BigTable caches these tablet locations for future use[5].

Summary:

Chubby file → Root tablet → Metadata tablet → user table

[5] 2006 - Bigtable - A Distributed Storage System for Structured Data (OSDI)