CSCI/ECEN 5673: Distributed Systems

Spring 2022

Homework 4

Due Date: 04/28/2022

Please submit a PDF copy of your answers via the submission link on Canvas by April 28, 2022. Write your answers in the space provided. Please DO NOT use any extra space. The space provided is sufficient for answering the questions.

Honor Code Pledge: On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance on this work.

Name: Sai Akhil Teja M

1. Both Dynamo and BigTable accepts a new server and makes it responsible for handling one part of the requests.
    a. How do both Dynamo and BigTable accepts a new server into the group?

In **Dynamo**, a gossip based protocol is used for propagating the membership and data partitioning changes among all the members. Upon a first-time join, node chooses its tokens which signify the positions in the ring.

In **Big Table**, whenever a tablet server is started, it creates and acquires a lock on a uniquely maintained file in a specific Chubby directory. The master monitors the server directory to discover the tablet server.

    b. Which portion of the data is assigned to this new server in BigTable and Dynamo?

In **Dynamo**, Distributed Hash Tables are used. These DHTs use consistent hashing and store key-values pairs. Just like in consistent hashing, whenever a new node joins, the keys of the neighbor nodes are transferred to this node based on the nodes' key-value pairs.

In **Big Table**, we have a 'servers' directory which contains the chubby locks of various tablet servers. It also has a master which tracks the assignment of tablets to tablet servers. This master queries all tablet servers with locks present in the servers directory. It then scans the metadata table to find all tablets and upon finding an unassigned table, the master finds a tablet server with sufficient room and assigns tablets to this server.

    c. Explain the assignment process and what happens when any client tries to access this data

For **Dynamo**, as discussed above, whenever a new node joins, some of the keys of the neighbor nodes are transferred to this node and the preference lists are updated accordingly. Each request shall be assigned a node and this node only serves requests if it's in the preference list. Else these requests are forwarded to the coordinator node in the preference list. A dynamo-aware client library is used to get a zero-hop distributed hash table. This client library needs to be updated with the new node information and it routes the requests directly to the appropriate coordinator node and clients can then access the data.

For **Big Table,** the master finds a tablet server with sufficient room for an unassigned tablet. Clients generally cache tablet locations, but a newly added tablet server may not be present. If the cache is empty, the client recurses up-heirarchy i.e. 3 round trips and a chubby query. In that case, it has to query chubby for the tablet location. Using this process, the client finds out the tablet server with a particular tablet location.

2. What are the key abstractions in Storm and how do they work together to process tasks?

- A storm application uses a topology of interfaces which create a ‘stream’ of transformations.
- Storm mainly has two key abstraction categories: 1. Core abstraction and 2. Top-level abstraction.
- Stream, Spout and Bolt - core abstractions.
- Top-level abstraction - Topology.
- A topology consists of sets of Spouts and Bolts, similar to a job in MapReduce. Storm uses topologies to do real time computation.
- A spout is a source of data/streams and a bolt processes data. Data is passed as tuples.
- Part of defining a topology is specifying for each bolt which streams it has to take as input. Whenever a spout or bolt emits a tuple to a stream, it sends the tuple to every bolt that subscribed to that stream.
- A stream grouping defines how that stream should be partitioned among the bolt's tasks. These groups organize tuples sent between two components.
- Shuffle grouping, field grouping and partial key grouping are few builtin streaming groups.
- Storm guarantees that every message flowing through a topology will be processed, even if a worker machine in the cluster fails.
- With Supervisor to control the state of a worker machine by starting worker as soon as a task is assigned to it and killed when decoupled from the supervisor.
- Nimbus daemon managed by Supervisor, is responsible for distributing code around the cluster, assigning tasks to machines and monitoring for failures.

3. (a) Traditional databases are based on tables with fixed rows representing different attributes of that record. Cassandra is a distributed database which is a “map of all maps”. Why do you think distributed databases deviate from the traditional database model? Discuss the advantages and disadvantages of both.

- The main motivation behind a distributed database is Scalability since they deal with different kinds of data and different schema unlike traditional databases.
- Data partitioning and Fault tolerance is easily achievable with a flexible schema which promotes Scalability.
- Most of the traditional databases have strong consistency but the distributed databases are okay with eventual consistency too however some support both.
- The read time is low for a traditional database due to efficient indexing whereas a distributed database has this tradeoff to achieve scalability.
- Traditional databases require caching layers where a distributed database don’t need any special caching layer.
- Traditional databases are good for reading/writing consistent data with lower latencies. Peer-to-peer hopping for data fetch might add extra latency to a read query for distributed databases.

b) Give an example of a query which is easier to represent in SQL than in Cassandra. Explain your answer.

- Any query that has a different structure of data from that defined in SQL is not easy to be represented in SQL whereas in Cassandra, it’s easy to manage structured, semi structured and non-structured data too due to it’s flexible schema.
- An example of such a query would be inserting a new record which contains a new field. This would not be possible as it would need a schema modification for SQL or any relational DBMS.

4. Read the following paper about Yahoo's PNUT database service:

B. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H. Jacobsen, N. Puz, D. Weaver and R. Yerneni. PNUTS: Yahoo!'s Hosted Data Serving Platform. VLDB 2008.

a) **[7 Points]** What are the motivations behind PNUTS design?
Scalability: The authors want not only architectural scalability, but also the ability to scale during periods of rapid growth by adding resources with minimal operational effort and minimal impact on system performance. (taken directly from paper).
Response Time and Geographic scope: The authors want the data platform to guarantee fast response times to geographically distributed users, even under rapidly changing load conditions brought on by flash crowds, denial of service attacks etc.
High Availability and Fault Tolerance: yahoo! applications must provide a high degree of availability, with application-specific trade-offs in the degree of fault-tolerance required and the degree of consistency that is deemed acceptable in the presence of faults.
Relaxed Consistency Guarantees: Most of the distributed replicated systems provide eventual consistency. Such consistency models are deemed as too weak and hence authors want to provide something better than this.

b) **[7 Points]** Describe the consistency model of PNUTS.
'*PNUTS uses a consistency model that offers applications transactional features but stops short of full serializability.'*
PNUTS provides a consistency model that is between the two extremes of general serializability and eventual consistency. It provides a per-record timeline consistency: all replicas of a given record apply all updates to the record in the same order. Most web applications modify one record at a time, whereas different records may have activity with different geographical locality. This was the main motivation behind choosing such a consistency model.

- For each of the record, one of the replicas is designated as the master and all updates for this record are directed to the master which then forwards this change to other replicas.
- **Read-any** can be used when there is no constraint on getting the up-to-date information.
- **Read-critical API** is used when it is needed to get the most up-to-date value.
- Such an API call may have **higher latency** but that's the tradeoff for getting the latest value.

c) **[11 Points]** What types of failures can PNUTS tolerate? Explain how PNUTS recovers from these failures.

- To propagate the changes to other replicas the masters publish the changes to the yahoo message broker (YMB), and these updates are then published to the same ymb which ensures that updates are always delivered in the commit order. The Yahoo Message Broker guarantees that messages are not lost, and published messages are delivered to all topic subscribers even if a broker fails.

- The collection of records are stored on server as a collection of tablets. There is a pretty flexible assignment of tablets to servers. Whenever a server fails, the recovered tablets are divided over multiple existing or new servers to spread the load evenly.
- Inorder to recover from failures, the controller first requests a copy from remote replica which then publishes a ‘checkpoint message’ into message broker to make sure any in-flight operations are executed at the source tablet and then the source tablet is copied to the destination.