CSE 512: Distributed Database Systems

Team Cassandrian

Topic: Data distribution plan

Since match data will be consumed by various match officials, organizations, or individual users i.e., viewers, we will be incorporating data distribution plan to minimize the latency when accessing the generated data.

Possible data distribution methods (could be changed as the project progresses):

1. Data replication:

   It minimizes the time and latency when performing read operations on database tables and accessing required information. Data replication also ensures that data is always available to the end-users and eliminates issues such as Single point of failure.

   Possible match tables that should be replicated:

   - Match: this table contains information about the match and will be replicated
   - Players: contains information about players involved in the match and their speciality
   - Bowl_by_bowl: this table will be containing information about every bowl that takes place during the match. This information will be match_id, bowl_no, over_no, among various other significant information.
   - Other tables that might be replicated are venue, team, match_officials, and match_player. These tables will not be having continuously changing elements as compared to above 3 tables, so data replication will not result in any substantial improvement in time and latency.

   Advantages of data replication are:

   - Improved Read Performance: Users can access information about teams and players from a nearby node, reducing latency.
   - Fault Tolerance: If one node goes down, users can still access team and player information from other nodes.
   - Load Balancing: The read load is distributed among multiple nodes, improving overall system performance.

2. Horizontal fragmentation:

   Since match data will be relatively huge in size, horizontal fragmentation of the table will result in efficient read operation.

   For match data, tables can be fragmented based on geographical locations of the venues, or teams.

   Possible fragmentations are as follows:

   - Fragmentation 1: Matches involving Team India
     - Match 1: Team India vs Team Australia
     - Match 2: Team India vs Team England... so on and so forth
   - Fragmentation 2: Matches involving Team Australia
     - Match 1: Australia vs Netherlands
     - Match 2: Australia vs Sri Lanka

Advantages of Horizontal fragmentations:

- Improved Query Performance: Horizontal fragmentation allows for parallel processing of queries. When queries are focused on a specific fragment, multiple fragments can be processed simultaneously, leading to improved query performance.
- Reduced Data Transfer Overhead: Since each fragment contains a subset of the data, queries that only require information from a specific fragment can minimize the amount of data transferred over the network. This reduces network traffic and improves overall system efficiency.
- Enhanced Scalability: Horizontal fragmentation facilitates the distribution of data across multiple servers or nodes. As the data grows, additional nodes can be added to the system, and the workload can be distributed, resulting in improved scalability.
- Increased Security: Horizontal fragmentation can enhance data security by restricting access to specific fragments. Access controls can be applied at the fragment level, ensuring that users or applications only have access to the data relevant to their needs.
- Customized Indexing and Optimization: Each fragment can be indexed and optimized independently based on the specific characteristics of the data it contains. This customization allows for more efficient storage structures and indexing strategies tailored to the nature of the data within each fragment.
- Support for Distributed Systems: Horizontal fragmentation aligns well with the principles of distributed systems, where data is distributed across multiple nodes. This makes it easier to design and implement distributed databases or systems that can scale horizontally.

3. Range-Based partition:
   Match data can be partitioned based on various ranges as required by the database designer.
   Few of the examples are as follows for match data:
   - Partition 1: Matches played between 2019-2023: this partition will display all the matches that were played between the year 2019 and 2023.
   - Partition 2: Matches played from 2015-2018.
   - partition 3: matches played from 2010-2014

   We can also partition based on runs scored by a batsman or wickets taken by a bowler. There are various other possible range partitions.

   Advantages of range-based partition are as follows:

   - Improved Query Performance: It can significantly enhance query performance, especially when queries involve ranges of data.
   - Parallel Processing: It enables parallel processing of queries and data operations. Different partitions can be processed concurrently by different nodes or servers, improving overall system throughput, and reducing query response times.
   - Scalability: As the dataset grows, range-based partitioning allows for easier scalability.
   - Better Load Balancing: It is possible to achieve better load balancing across nodes. If the data is evenly distributed based on a logical criterion like date ranges, each node can handle a similar workload, preventing hotspots and ensuring a more balanced system.
   - Ease of Data Retrieval for Time-Dependent Queries: It is particularly beneficial for time-dependent queries, where users are interested in data within specific time intervals. It allows for quick isolation of the relevant partition, leading to faster query response times.