Tcaplus database

Best Practice

Product Documentation

Best Practice

Last updated：2019-02-13 10:02:46

Best Practices for Table Definition

TDR table definition

1. The key field used by the primarykey requires high dispersion, so that it can be distributed to multiple nodes at the access layer when the gameserver sends a request.
2. The key field used by the splittablekey requires high dispersion, and the range of selected values cannot be concentrated in a limited set.
3. In case of table definition, the index key cannot be the same as the primary key; otherwise, it is a waste of network and disk resources.
4. In case of array definition, the refer attribute (count is the defined size, and refer is the actual size) should be added to the value fields, so that the count size can be expanded later, and the data footprint in network transmission and on disks can be reduced.
5. The value fields should be primary fields to reduce nesting between fields. The nested depth is limited to 3.
6. It is not recommended to use the binary type for the key fields as it hinders follow up efforts.
7. The number of indexes of a single table definition is limited to 8. It is recommended to use 2 to 3 indexes, which should be set as needed.

Protocol Buffers table definition

1. The key field used by the primarykey requires high dispersion, so that it can be distributed to multiple nodes at the access layer when the gameserver sends a request.
2. The key field used by the splittablekey requires high dispersion, and the range of selected values cannot be concentrated in a limited set.
3. Defining `non-primary key` fields with a type of nested structure is supported. The maximum nested depth is restricted to 30, and the data access capacity will be compromised if this limit is exceeded.

Best Practices for gameserver

1. In scenarios involving reading or writing of partial fields, it is recommended to read and update partial fields, so as to reduce the data size for network transmission and the number of disk reads and writes, thus avoiding invalid data fetching.

2. If the data record needs to be returned when a writing operation is performed, it is recommended to set result_flag as needed. For example, if the updated data record is required after an update operation is performed, the result_flag should be set to 2. If the data record is not required during a writing operation, the result_flag is set to 0.
3. For the command words in batch processing tasks, such as batchget, listgetall, and getbypartkey, it is recommended to get the data record based on offest and limit. We recommend setting the "limit" to 200, and multi-package return is enabled at the gameserver side.
4. For list-related tables, when a listaddafter operation is performed, a phase-out rule should be set in case that the list is full. It is recommended for the rule to add a new record to the front and delete a record from the rear, or vise versa.
5. For list-related tables, a correct index should be passed for listreplace, listdelete, and listbatchdelete operations.
6. Before the gameserver reads data, make sure that the value fields of the data record obtained is not empty. For example, when there are three value fields: A, B and C, if the gameserver only got A and B, then the C field is empty. Please proceed with caution.
7. The gameserver should control the timeout locally, and it is not recommended to determine the timeout based on the response package sent from the gameserver.
8. When the gameserver performs traversal, it is recommended to get the data from the nodes at the storage layer, so as to avoid affecting the performance of the primary node.
9. It is not recommended to perform memset operation on large fields to avoid consuming CPU resources. Set the values of some fields for large data structures to 0 before initialization.
10. When processing requests to Tcaplus and responses from Tcaplus, the gameserver should use the divide-and-conquer method to process partial requests sent to Tcaplus first and then process responses returned from Tcaplus.

Best Practices for System Design

1. Create separate tables for the frequently used fields and the fields on which an one-time atomic operation needs to be performed.
2. When the gameserver writes data back, it is recommended to control traffic and discretize the data by time.
3. Dynamic modification of table structure is supported, and a table data conversion plugin is provided.
4. Rollback to the point in time of your cold backup or to an exact time at the table/logging level in all-server/all-region mode is supported.
5. Nodes at the access layer and the storage layer can be dynamically scaled for both all-server/all-region mode and multi-server/multi-region mode. We recommend the multi-server/multi-region mode.

6. Fields with a logical relation should be merged into one table to avoid distributed transaction problems.
7. It is recommended to enable the compression feature, including the compression of request/response packages and logs.