Project 2: Tribbler

Xi Yao (xyao1) Xinyan Wu (xinyanw)

tribserver

design:

Receive a request from client, forward it to libstore and respond according to the result get from libstore.
In GetTribbles/GetTribblesBySubscription function, the result retrieve from libstore is not guarantee to be ordered, so need to be sorted (with sort package) before return.
In PostTribble function, to avoid the race condition that two or more servers receive PostTribble calls for the same user at the same time, it should check if server generate a unique postkey before call functions in storage system to store the tribble.
In order to store tribbles for a user efficiently, firstly maintains tribbleListKey/postkey_list pair and then for each tribble, maintains postkey/tribble pair.

libstore:

data structure

libstore struct:
mode: LeaseMode(never/normal/always)
client: the rpc client of the master node
servers: a list of all server nodes(include master and slave node)
myHostPort: libstore host post
cache: a map to store key/cacheValue pair
queries: a map to store every query timestamp in QueryCacheSeconds for every key
cachelock: to lock cache map
querylock: to lock query map

cacheValue struct:
leaseEnd: a cache is valid before leaseEnd
value: use it to store if the cached value is key/value pair, or is nil
list_value: use it to store if the cached value is key/value_list pair, or is nil
valid: if revoked set to false, default true

synchronization

1. Use a mutex lock to avoid concurrent visit of R/W of caches stored in the libstore

2. Also use a mutex lock to avoid concurrent visit of R/W of query timestamp stored in the libstore

algorithm

1. Request routine:

Sort all server nodes by their node id, and assign to first node bigger than the hash, if can't find, assign to first node.

2. Cache and query timestamp:

Once get a get/getlist query, it should add this query's timestamp in the queries, and clean invalid query timestamp in corresponding list. Then visit the cache to find if valid stored in the cache. If valid, just return the value to tribserver. If not, then decide if is need to send wantLease to storage server by count the queries list length. Once get reply from storage server and is granted a lease, add it to the cache.
Go a routine to clean cache by delete invalid and timeout caches.

3. Revoke Lease:

Revokelease function implements a RevokeLease RPC callback. If the storageserver call this function , it will set cached value to be invalid, waiting to be delete.

Storageserver:

data structure

storage server structure:
servers: a slice used to store server nodes
kvStore: the map of which the key is args.Key, the value is wrapped item: myItem
listMap: the map of which the key is args.Key, the value is wrapped item list: myItemList
dialMap: a map used to store the clients that hold leases
lockMap: a map to store per key lock
allRegisterDone: channel to indicate that all slave servers have join the ring

Algorithm

1. Lease :
Since each key maintain a leaseMap, it can keep track of what leases it has granted. Each time libstore want a lease in Get or GetList, we will first check if the revoke function called for the key, if does, storage server will reject the wantless request. If not, the leaseMap for that key will new a lease by add a new start time and hostport pair to the leaseMap.

2. Modify function(Put, Delete, Append, Remove):
When libstore request a modify request, storage server will first use the key to check function whether the server is right, and then will check whether the key is in map. When libstore send a modify request, we will first check whether the key has granted leases for some clients before, if does we will first revoke the leases, and clear the leaseMap for the key, after that operate the modification.

Synchronize

For servers, we have a serversLock.
For Put, Delete, Get, Append, Remove, Getlist function, We use per key lock here. We have a lockMap to store lock for each key. Each time enter those function, we first check if there is a lock for the key, if no lock for this key, new one and store it. Otherwise use the stored lock directly.

Division of work:

Xi Yao(xyao1): tribserver.impl and libstore.impl

Xinyan Wu(xinyanw): storageserver.impl