MEP 23 -- Multiple memory replication design

Current state: Under Discussion

ISSUE:

PRs:

Keywords: Segment, QueryNode, QueryCoord, Multiple memory replications

Released:

Summary

We want to add a new feature that maintain multiple segment replications in memory to improve the QPS and support fast failover.

Motivation

Milvus holds a replication per segment in memory for now. To improve the search request concurrency, Milvus can have multiple replications of a segment on different QueryNodes. If a QueryNode is doing a search request on a segment and another request arrives, we can send this request to another QueryNode which have a replication of the same segment. Besides, if a QueryNode crashes, now we have to wait for a segment being loaded to search. If we have multiple replications, we can resend the search request to another QueryNode immediately.

Public Interfaces

No public interfaces need to be added or changed. It's transparent to users.

Design Details

- Balance
  - If a new segment need to be loaded, we will try to allocate it to multiple different QueryNode(<=replication number). It is not necessary to wait all replcas are loaded successfully bacause a replication can support the search request.

- Search
  - Cache
    - Proxy maintains a cache mapping **segments to QueryNodes** which is updated periodically. When Proxy receives a request, milvus will get all sealed segments needed to be searched from cache and try to assign them to QueryNodes evenly.
    - And for growing segments, Proxy also maintain a **channels to QueryNodes** cache and send request to corresponding QueryNode.
  - Failover
    - Because of the inaccuracy of the cache on Proxy, some segments/channels may have been moved to other QueryNodes at that moment. In this case, Proxy will receive response with error, update the cache and try to assign it to another QueryNode
    - If Proxy can not find this segment after updating the cache(eg. compaction happened), it will ignore this segment.
    - Proxy may miss some segments because the cache is accurate. QueryNodes with **DMLchannels(growing segments)** will return search response with the reliable segments list at this moment. Proxy will compare it with local cache. If Proxy actually miss some segments, it will update the cache and search missed segments.
  - Enhancement
    - Proxy can not allocate search tasks to QueryNode completely evenly and QueryNodes may have different resources to serve search requests. To avoid the long tail effect, if a QueryNode is idle and it have segments needed to be searched on other QueryNodes, prxoy will assign these segments to it.

Test Plan