MuCache源码剖析

An example of microservice(Movie)

Services(app)

> example:
>
> 1. `service(app)`: [key]
>   - (methods)
>   - endpoint1
>   - ro_endpoint2
>   - (call other services' endpoint...)
>   - [service_id]-endpoint

1. `cast_info`: ['cast_id']
    - store_cast_info
    - ro_read_cast_infos
2. `compose_review`: ['review']
    - compose_review
        - 9-store_review
        - 6-upload_movie_review
        - 12-upload_user_review
3. `frontend`: ['ComposeRequest']
    - compose
        - 11-login
        - 10-get_unique_id
        - 11-ro_get_user_id
        - 4-ro_get_movie_id
        - 2-compose_review
4. `movie_id`: ['title']
    - register_movie_id
    - ro_get_movie_id
5. `movie_info`: ['movie_id']
    - upload_movie_info
    - store_movie_info
    - ro_read_movie_info
6. `movie_reviews`: ['movie_id']
    - upload_movie_review
    - ro_read_movie_reviews
        - 9-ro_read_reviews
7. `page`: ['movie_id']
    - ro_read_page

        - 5-ro_read_movie_info
        - 1-ro_read_cast_infos
        - 8-ro_read_plot
        - 6-ro_read_movie_reviews
8. `plot`: ['plot_id']
    - write_plot
    - ro_read_plot
9. `review_storage`: ['review_ids']
    - store_review
    - ro_read_reviews
10. `unique_id`: ['req_id']
    - get_unique_id
11. `user`: ['username', 'password']
    - register_user
    - login
    - ro_get_user_id
12. `user_reviews`: ['user_id']
    - upload_user_review
    - ro_read_user_reviews
        - 9-ro_read_reviews

通信方式

- ZeroMQ（between wrappers andthe cache manager）
    - fullReq -> WQ(channel)
        - fullReq["type"] = "S/E/I/IC/SC"
        - fullReq["inner"] = request
- Http（between cache managers）

Cache Manager(CM)

- Cache
    - 关键参数ca的定义：ca = cm.CallArgs(wrappers.HashCallArgs(app, method, buf))
    - CacheGet：通过ca索引获取缓存结果
    - CacheSet：键值对（key为ca）
    - CacheRemoveCalls：删除一系列calls对应的键值对
    - CacheSaveCalls：保存一系列键值对{ca：ReturnVal}
- cm
    - 所有参数都转化为String类型
    - TypeStartRequest = "S"
    - TypeEndRequest = "E"
    - TypeInvRequest = "I"
    - TypeInvCallsRequest = "IC"
    - TypeSaveCallsRequest = "SC"
- conn：ZeroMQ、http通信
- deser：各种数据类型的转换和处理

- **state**
  - type HistoryItem interface {

    IsCallArgSet() bool

    IsWriteKey() bool

    IsInvCall() bool

    }
  - state
    - history []HistoryItem（Contains a sequence of started calls and key writes）
    - keyDeps expirable.LRU[cm.Key, CallsAndCallers]
    - callDeps expirable.LRU[cm.CallArgs, CallsAndCallers]
- **server**
  - sendSaveToCaller
    - request：CallArgsList + ReturnVals
    - caller
  - sendInvsToCallers：saved（Dict map[cm.ServiceName]map[cm.CallArgs]struct{}）
  - **process：处理5种操作**
    - StartRequest：将call加入history切片（保证顺序性）
    - EndRequest：先检查isvalid
      - valid：发送Save信号 + 存储keyDeps/callDeps
      - invalid：不处理
    - InvalidateKeyRequest：先检查是从哪里收到的inv message
      - 从其他的CM：将invKey加入history（IsWriteKey() ） + 从keyDeps中删除（如果存在，则通知其他的caller）
      - 有切片且不是从其他切片的CM获得的消息：获取所有neighbours，然后向他们发送inv message
    - InvalidateCallsRequest：将invCalls加入history（IsInvCall() ） + 从callDeps中删除（如果存在，则通知其他的caller）+ 从cache中删除键值对
    - SaveCallsRequest：给cache发送save message，存储ca对应的键值对

Wrapper

- Context
  - "caller"
  - "RID"（Call_Id）
  - "read-only"
  - "call-args"
- Deps
  - 表示一个结构体{Call_Id：idDeps}
  - `idDeps` 是一个 `sync.Map` 类型的变量（其中key类型为cm.Key / cm.CallArgs，value字段为空结构体）
- Wrapper
  - PreReqStart：对callid初始化deps，将callArgs构造成指定格式发送到WQ(work queue)
  - PreRead：写入新的依赖，deps.AddKeyDep(callId, key)
  - PreWrite：将key构造成invalid request发送到WQ

    - PostWrite：与PreWrite重复，完全一样
    - PreReqEnd：构造endReq := cm.EndRequest{CallArgs: callArgs, KeyDeps: keyDeps, CallDeps: callDeps, Caller: currServiceName, ReturnVal: retVal}
    - PreCall：拓展了PreRead，在cache中查找key-value
    - ROWrapper / NonROWrapper：泛型函数，包装了具体的method函数

State

- Write to DB
    - SetState：单个写入
    - SetBulkState：批量写入
- Read from DB
    - GetState：单个读取
    - GetBulkState：批量读取

Invoke

- SaveArgs结构体 -> SavingQueue
    - Ca（cm.CallArgs）
    - RetVal（cm.ReturnVal）
- performRequest：发送request并获取回复信息，解析后存入Cache（通过SavingQueue）
- Invoke：如果cache命中则直接返回请求结果；如果未命中再调用performRequest发送请求