1. 为什么使用Redis

Redis和Memcached做为缓存，读写性能差距不大，测试结果都在7W+条/秒。与Memcached相比，使用Redis主要基于以下三个原因：

1. Memcached内存管理采用固定分配的方式（Chunk），比较容易浪费内存空间
2. Memcached使用LRU淘汰数据，数据超过内存上限时会被清除。
3. Memcached仅支持简单的Key-Value数据结构
4. 整体介绍

支持分布式缓存，各节点的配置在全局数据库的t\_cache\_config表中。分为全局缓存和

玩家缓存两部分。

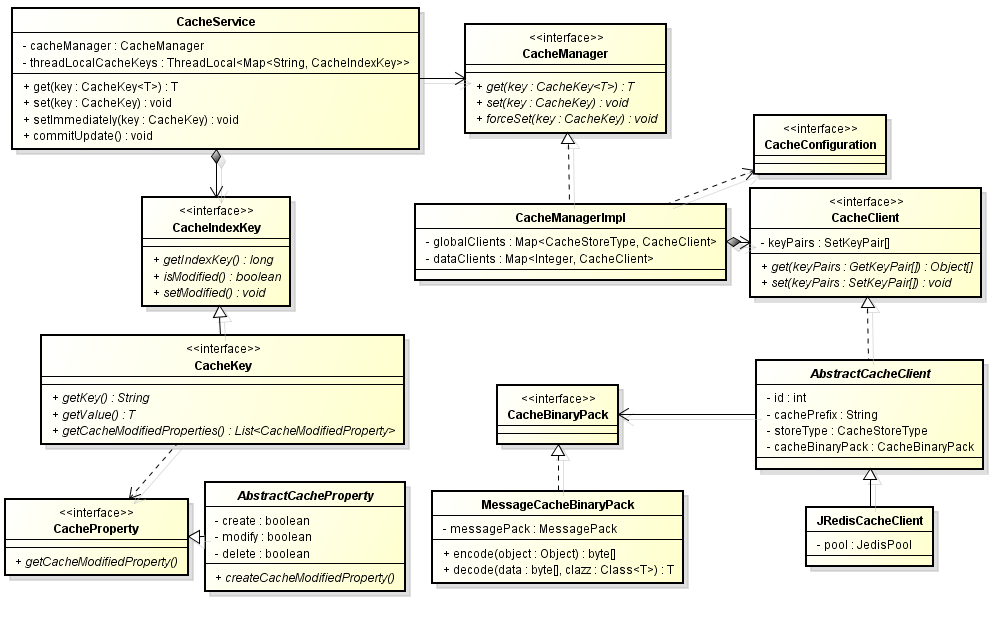
全局缓存包括索引、离线玩家索引、全局数据、聊天四种。索引缓存主要用于以下情况：

|  |  |  |
| --- | --- | --- |
| Key | Value类型 | 说明 |
| PLAYER\_UNIQUE\_KEY[PID] | String | 用户唯一认证Key |
| PLAYER\_ID\_AUTO\_INCREMENT | Long | 玩家ID自增 |
| ROLE\_ID\_AUTO\_INCREMENT | Long | 角色ID自增 |
| LOAD\_BALANCE\_SERVER | SortedSet | 服务器负载情况 |
| LOAD\_BALANCE\_CACHE | SortedSet | 缓存负载情况 |
| LOAD\_BALANCE\_DATABASE | SortedSet | 数据库负载情况 |
| PLAYERS\_INDEX | Map | 玩家索引，记录玩家的服务器ID、缓存ID、数据库ID |

离线玩家索引缓存，用于战斗搜索；全局数据缓存用于全局性业务，比如排行榜、帮会等；聊天缓存用于聊天业务。

玩家数据缓存用于存储玩家的数据，不参与业务计算。

1. 核心类图及说明



CacheService包含本地ThreadLocal缓存和CacheManager。CacheManager包含多个CacheClient，并通过CacheClient对缓存进行操作。CacheClient封装对于缓存的相关操作，例如set，get方法，直接操作缓存。

CacheKey是对需要缓存的数据的key的封装，CacheProperty是对数据变化标记的封装，完成一次数据缓存修改(增删改)的同时会在缓存中添加相应的数据变化标记，DBS会根据缓存中的数据变化标记将缓存数据定期同步入库。

CacheConfiguration是缓存配置参数，CacheBinaryPack是数据序列化工具，MessageCacheBinaryPack实现了MessagePack的序列化方式，相比Java自己的序列化方式，该方式速度更快，空间更小。

1. 玩家数据读取过程

从缓存中得到玩家索引

存在

在本地

缓存中创建玩家索引

在缓存

还原成内存对象

保存到本地

取出

从数据库加载

保存到缓存

绿箭头表示是

红箭头表示否

具体参见CacheService的get方法

1. 玩家数据存储过程

从缓存中得到玩家索引

实时

保存到本地

清除本地

保存到缓存

获取本地数据

保存到缓存

绿箭头表示是

红箭头表示否

**保存过程**

**提交过程**

具体参见CacheService的set、setImmediately、commitUpdate方法