诛仙实现

需求

* top n周期榜
* top n实时榜
* 全排行帮
* 个人排名获取

思路

top n周期榜

老诛仙方式

top n实时榜

* 不应更新太频繁
* 周期存盘

存储压力可控，但也可能造成数据丢失

* 不保证数据完全准确

关闭服务器和程序崩溃导致部分数据丢失

score变小掉出排行榜导致尾部数据不准确

* 定时重建

停服期间重刷，获取正确数据

全排行实时榜

数据量大的话，考虑交给Redis去做排行，但额外需要一或多台机器

数据量小在程序内实现，设想是这样：程序启动时从所有数据中筛选出活跃数据，对活跃数据进行排名，该数据不存盘

个人排名获取

上榜是准确排名，未上榜是近似排名

计数排序，离线获取排名分布，桶按分数以特定规则划分（最好是根据实际情况使用不同规则）

其他问题

* 冷启动问题

开始数据量少的时候，如何保证个人排名准确性

* 数据一致性

实时榜如何与数据库数据保持一致

宕机或进程被杀掉如何保证数据的正确性

score减小掉出排行榜的情况很难保证排行榜的完全正确

1.预写日志，崩溃后程序启动时从日志恢复，实现复杂，但能保证榜单立即恢复正确

2.停服期间刷一次榜，获取最新榜单，实现简单，但可能要在下次维护之后榜单才能完全恢复正确

* 分页传输

如实时榜在内存中不是有序的，如何分页传输

* 存盘

一个排行榜存为一个value

避免频繁存盘

* 重复问题

如以score为key排序，会出现重复的key

top n实时榜方案

score递增的榜单

方案一

struct RankInfo

{

int key;

int score;

int other\_info;

};

map<int, RankInfo> rankMap;//<score+key,RankInfo>

hashmap<int, int> scoreMap;//<key,score>

方案二

排序交给客户端，或者服务器定时（比如5分钟一次）进行排序。

struct RankInfo

{

int key;

int score;

int other\_info;

};

RankInfo rank[N];

RankInfo \*min;

hashmap<int, RankInfo\*> rankMap;

一个对象的score增大了，主要操作有：

查询该对象在榜单中的信息

该对象已在榜单中，更新其score

该对象不在榜单中，榜单未满，将该对象加入榜单

该对象不在榜单中，榜单已满，该对象上榜，替换掉另一个对象

方案对比

|  |  |  |
| --- | --- | --- |
|  | 方案1 | 方案2 |
| 查询（最频繁的操作） | O(logN) | O(1) |
| 更新 | 删掉旧分，添加新分 3O(logN) | O(1) or O(N)(update min) |
| 新加入 | 2O(logN) | O(1) |
| 替换 | 3O(logN) | O(N)(update min) |

倾向于选择方案二，查询效率高

榜单频繁变化问题

榜单不能更新太频繁，比如等级排行榜，可能俩人经验差不多，打一只怪两人排名就会发生变化，可以考虑实时榜用等级排名，离线刷的时候用经验值排名。

另外服务器可以记一个上次更新时间，客户端记一个上次获取榜单的时间并缓存榜单，客户端下次获取榜单时如未发生变化则无需再次传输数据，对于变化不频繁的排行榜有用。

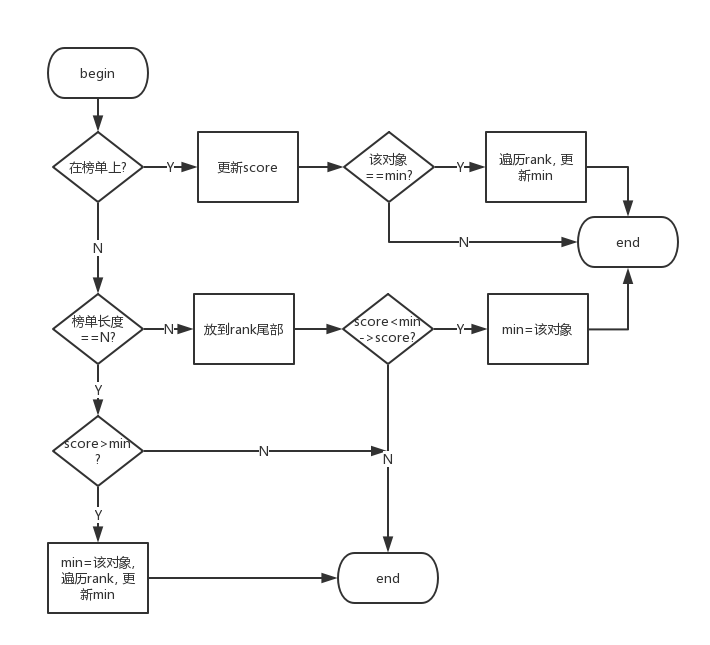
score可变小的榜单

出现成员掉出榜单需要补位的情况，不可能去刷一遍数据库，可能会导致尾部数据不正确。

建议离线刷。

流程

方案二score递增：



方案二score可变小：

min==该对象 score变大 遍历rank，更新min

min==该对象 score变小 更新score

min!=该对象 score变大 不处理

min!=该对象 score变小 1比min->score小 min=该对象 2否则不处理

全排行实时榜方案

程序启动时遍历一遍数据库，从中筛选出活跃对象，组成实时榜，该榜不存盘。