Top n排行榜

问题

* 实时/离线

离线实现比较简单

* 数据一致性

实时榜如何与数据库数据保持一致，宕机如何保证数据的正确性

* score递增/可以变小

score如果可以减小，掉出排行榜的情况很难保证排行榜的完全正确

* 服务器实时排序

如果服务器实时（按score）排序，排名数据更新较频繁的情况可能负担很重，客户端可以分页传输 ，容易获得个人名次

如果服务器存储的是不排序的榜单，排序交给客户端来做，则无法分页传输，流量会增多，榜单变化时可能需要顺序遍历才能更新排名最小者

方案

score递增的榜单

方案一

struct RankInfo

{

string name;

int score;

int other\_info;

};

multimap<score, RankInfo> rankMap;

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

{

string name;

int score;

int other\_info;

};

RankInfo rank[N];

RankInfo \*min;

hashmap<string, RankInfo\*> nameMap;

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+M) | O(1) |
| 更新 | O(1) | O(1) or O(N)(update min) |
| 新加入 | O(logN) | O(1) |
| 替换 | O(logN) | O(N)(update min) |
| 内存 | 小 | 大 |
| 复杂度 | 低 | 高 |

倾向于选择方案二，查询效率高

其他问题

数据一致性问题

不应当频繁存盘，可以定时存盘，但宕线可能会导致数据不一致，解决方法有两种：

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

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。

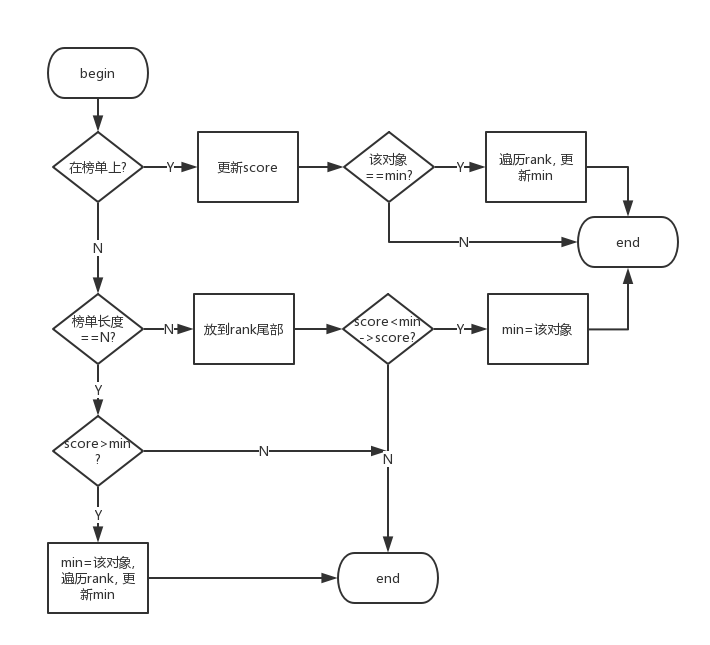
另外服务器可以记一个上次更新时间，客户端记一个上次获取榜单的时间并缓存榜单，客户端下次获取榜单时如未发生变化则无需再次传输数据，对于变化不频繁的排行榜有用。

score可变小的榜单

建议离线刷。否则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。

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