活动排行榜系统

需求描述

你开发了一个游戏，日活跃用户在10万人以上。请设计一个活动排行榜系统。

- 在每月活动中，玩家得到的活动总分为0 到10000 之间的整数。
- 在每月活动结束之后，需要依据这一活动总分，从高到低为玩家建立排行榜。
- 如果多位玩家分数相同，则按得到指定分数顺序排序，先得到的玩家排在前面。
- 系统提供玩家名次查询接口，玩家能够查询自己名次前后10位玩家的分数和名次。
- 请使用UML图或线框图表达设计，关键算法可使用流程图或伪代码表达。

备注：请根据个人技术特长，编程语言不限，优先使用Go。如无不便，请优先使用github.com，gitlab.com 或gitee.com 提交答案，回复公开代码库地址即可。否则，请使用zip 压缩包以邮件附件形式提交代码。

需求分析

需求提取

- 活动排行榜系统
- 日活用户超过10w
- 每月活动结束，建立排行榜
- 查询前后相邻10名的玩家分数，名次
- 记录用户分数变化

QPS预估

1. 假设日活用户为 20w
2. 玩家写入分析
    a. 假设每个用户都到达最高10,000分
    b. 上限次数为 10,000/30 = 333
    c. 平均TPS = 20w * 333 / 86400 = 770
    d. 假设峰值是平均的3倍，峰值TPS= 770 * 3 = 2310

3. 玩家查询分析
    a. 活动结束后，用户名次固定，所以查询后，可以缓存结果。所以压力都在第一次查询。
    b. 平均QPS = 20w / 86400 = 2.3
    c. 由于活动特殊性，假设按照二八原则，80%用户在20%时间请求。
    d. 假设峰值的是平均10倍
    e. 峰值QPS = 20w * 10 * 0.8 / (86400 * 0.2) = 92
    f. 所以第一次获取名次的查询，可以使用mysql处理

存储分析

1. 根据qps分析，峰值TPS 2,310，超过了mysql 普通的写入性能，后续会成为系统瓶颈
2. 引入消息队列，起到削峰填谷的作用，将峰值TPS，降到平均的TPS 770左右，结果批量写入，就能解决存储访问压力
3. 每天写入数据的峰值 20w* 333， 所以随着活动运营天数增加，系统存储数据量，也不断膨胀
4. 通过分表方案，提升数据库查询性能。

系统设计

接口设计

1. 写入接口 POST /score/create
    a. 当获得score时，提交
    b. 参数
        i. activity_id
        ii. score
        iii. token（ token 默认能识别出user_id等信息)
    c. 返回
        i. success
2. 查询接口 GET /rank/nearest
    a. 参数
        i. activity_id
        ii. token （ token 默认能识别出user_id等信息)
    b. 返回
        i. 排名列表

流程图：

见最后

设计思路：

- 玩家提交分数逻辑设计：
  - 玩家每次获得分数上报服务端
  - 服务端保存玩家上报分数明细
  - 验证请求合法后，先写入消息队列
  - 服务端异步处理消息任务，写入MySql
- 玩家名次查询逻辑设计：
  - 因为活动到期后，排行榜名次固定
  - 为了提升接口性能，可以采用客户端和服务端，同时缓存排行榜数据
  - 建议缓存时间不宜过长
  - 服务端后续接到请求，检查数据是否有更新
  - 服务端按玩家id为key 缓存返回结果
- 服务端建立活动排行榜设计：
  - 通过qps 预估分析，玩家上报分数明细数据量大
  - 先按天统计前一天的得分和排名
  - 活动结束后，根据按天结算，统计整个活动排名
  - 这样就优化了总排名生成时间和压力

数据表设计

表格设计

活动表(activity_info)

id	title	start_date	end_date
1	xx活动	2023-08-01 00:00:00	2023-08-31 00:00:00

activity_info id 为主键 按id 查询某个活动

活动排行榜表(activity_rank)

id	activity_id	user_id	score	rank	updated_at
1	1	1	10000	10	2023-08-13 00:00:00
2	1	2	8000	15	2023-08-13 01:00:00

注：rank 单个活动值不重复

activity_rank (activity_id, user_id) 为唯一键 优化查询单个用户的名次，保证用户在活动里唯一

activity_rank (activity_id, rank) 为键 优化玩家名次查询

活动排行榜日表(activity_rank_day)

id	activity_id	user_id	score	rank	updated_at
1	1	1	10000	67	2023-08-13 00:00:00
2	1	1	8000	90	2023-08-14 01:00:00

注：增加日榜提升活动排行榜运算速度； rank 单个活动值不重复

activity_rank_day (activity_id, user_id) 为键 优化服务端建立活动排行榜按activity_id 和 user_id 组合查询

玩家得分表(user_score_xx)

id	user_id	activity_id	score	type	创建时间
1	1	1	1	1	2023-08-01 00:00:00
2	1	1	2	2	2023-08-02 08:00:00
3	3	1	1	1	2023-08-01 00:10:00

注：user_score_xx xx为活动日期进行的天数；type 玩家得分来源，方便检查用户是否作弊

user_score_xx (activity_id, user_id）为键 优化服务端建立活动排行榜按activity_id 和 user_id 组合查询

主要逻辑伪码

存储玩家玩家提交分数

select * from activity_info where id = x ; #检查活动是否在有效期内

insert into user_score_xx (user_id, activity_id, score, created_at) value (x,x,x,x)

服务端建立活动排行榜

- 每日统计数据同步到活动排行榜日表

select user_id, activity_id, sum(score), max(created_at) from user_score_xx where activity_id=x and user_id > xx group by activity_id, user_id order by user_id asc limit 1000

batch insert into activity_rank_day

update rank

- 活动结束后从活动排行榜日表统计数据存入活动排行榜月榜

select user_id, activity_id, sum(score), max(updated_at) from activity_rank_day where activity_id=x and user_id > xx group by activity_id, user_id order by user_id asc limit 1000

batch insert into activity_rank

update rank

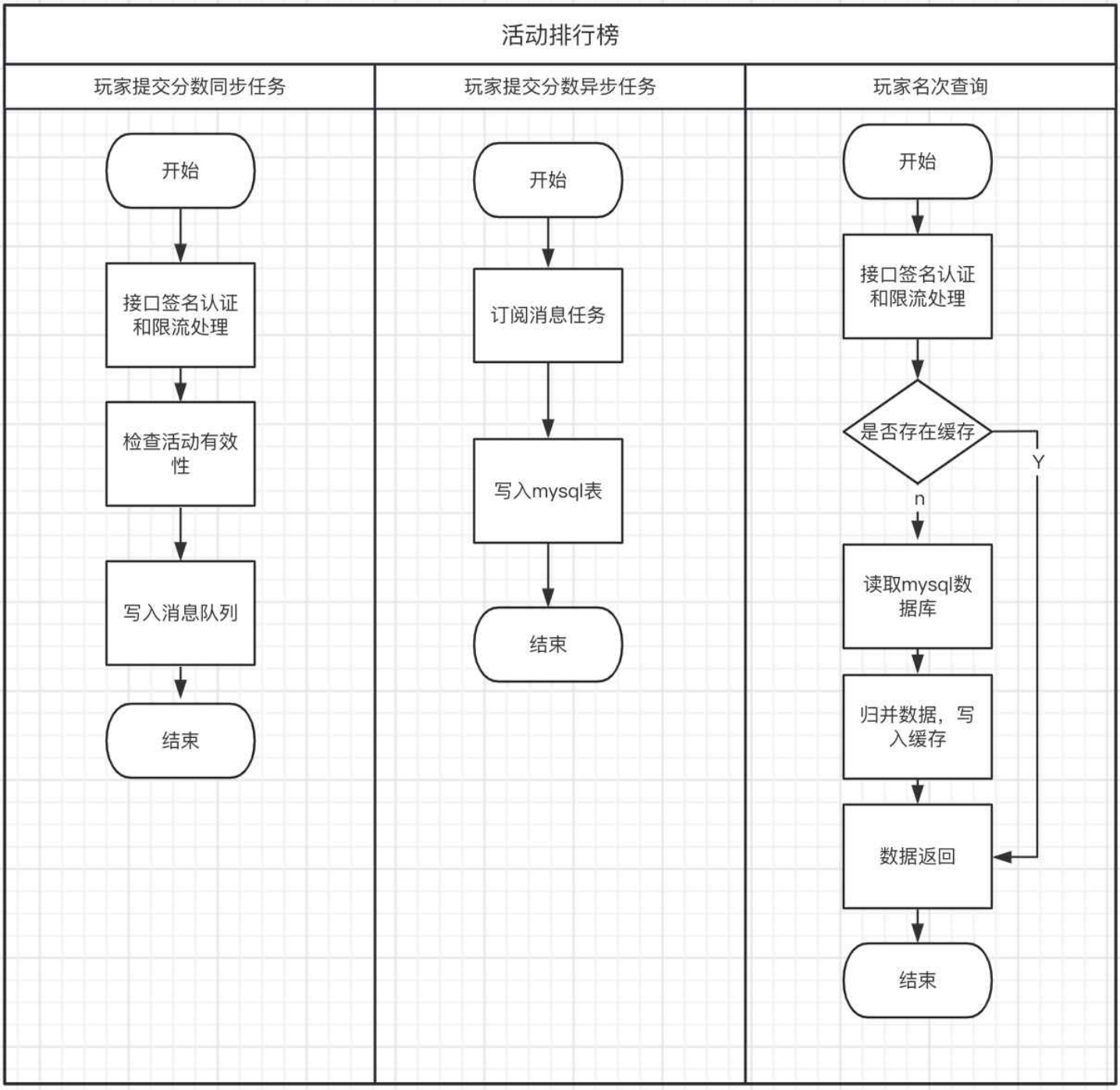
玩家名次查询

select user_id,score, rank from activity_rank where activity_id=x and user_id = x

select user_id,score, rank from activity_rank where activity_id=x and rank < x limit 10

select user_id,score, rank from activity_rank where activity_id=x and rank > x limit 10

流程图：


活动排行榜
玩家提交分数同步任务
开始
接口签名认证和限流处理
检查活动有效性
写入消息队列
结束
玩家提交分数异步任务
开始
订阅消息任务
写入mysql表
结束
玩家名次查询
开始
接口签名认证和限流处理
是否存在缓存
Y
n
读取mysql数据库
归并数据，写入缓存
数据返回
结束