**13-5｜红包雨底层逻辑实现**

**底层逻辑注意事项**

**生成红包逻辑**

* 红包数据的存储位置如何设计？

临时性数据，持久化存在MySQL，内存当中，读写性能非常好。通常做法是：本地内存存储红包雨数据 或者 redis存储红包雨数据。

主播id -》红包雨配置（唯一的code）

1. 使用Redis的List集合存储红包数据（key包含code）-> pop接口领取红包（单个List存储1k个红包）（redis的分片存储，100个list）
2. 本地内存存储红包数据。（500台tomcat，每台tomcat的本地内存里存储一定量的红包数据。20000个红包，一共是1000w个红包，2亿元红包，1kw人瓜分）（结合dns访问所在地最近的一台clb，clb-> nginx集群 ->gateway

-> tomcat, 500台tomcat，假设其中的某一台挂了）

* 红包的领取是怎么一个过程？

Redis + List存储红包数据（每场直播对应一个List，1000个红包）

* 如果按照主播id为维度去生成红包数据是怎么存储？

List（一定要有一个唯一的标识，key最好有一个唯一的code识别）

A主播开了一场直播，生成了一场红包雨的数据，但是没有领完，然后下播了。此时A主播下播后立马又获取了下一场直播的红包雨奖励。

list:(id) -> (上一场没有领完的红包数据)

list:(id) -> (往旧的list集合中塞入了红包数据)

* 生成红包的流程是怎样的？

主播点击前端的一个按钮去生成的。

* 如何防止红包重复生成？
* 如何保证红包生成的均匀性？如何防止不超出预期金额限制？

**抽红包逻辑**

* 领取红包的流程如何设计？（redis的List的pop接口领取红包，如果是本地内存的话，queue也有pop功能）
* 如何防止恶意刷接口调用领取红包？（A直播间，恶意请求可以领取到B，C，D等多个直播间的数据）
* 如何通知全场观众开始参与抢红包行为？（点击生成红包按钮，im异步通知主播，红包初始化完成，主播才有权限点击下一步，开始抢红包）
* 直播间之间的领取红包接口不能直接互通，如何考虑接口安全性？

**生成红包的实现**

|  |
| --- |
| Java /\*\*  \* 生成红包金额  \*  \* @param totalCount  \* @param totalPrice  \* @return  \*/ private static List<Integer> createRedPacketPriceList(Integer totalCount, Integer totalPrice) {  List<Integer> redPacketPriceList = new ArrayList<>();  for (int i = 0; i < totalCount; i++) {  int maxLimit = (totalPrice / (totalCount - i)) \* 2;  int currentPrice = ThreadLocalRandom.current().nextInt(1, maxLimit);  //最后一个红包  if (totalCount == i + 1) {  redPacketPriceList.add(totalPrice);  break;  }  totalPrice -= currentPrice;  redPacketPriceList.add(currentPrice);  }  return redPacketPriceList; } |

* 拆解List集合工具方法

|  |
| --- |
| Java /\*\*  \* 将一个List集合拆解为多个子List集合  \*  \* @param list  \* @param subNum  \* @return  \* @param <T>  \*/ public static <T> List<List<T>> splistList(List<T> list, int subNum) {  List<List<T>> tNewList = new ArrayList<>();  int priIndex = 0;  int lastPriIndex = 0;  int insertTimes = list.size() / subNum;  List<T> subList;  for (int i = 0; i <= insertTimes; i++) {  priIndex = subNum \* i;  lastPriIndex = priIndex + subNum;  if (i == insertTimes) {  subList = list.subList(priIndex, list.size());  } else {  subList = list.subList(priIndex, lastPriIndex);  }  if (subList.size() > 0) {  tNewList.add(subList);  }  }  return tNewList; } |