系统设计面试真题 | 设计Ticket Master

原创 2016-03-29 东邪老师 九章算法

学员面经题

一个类似ticket master的网站。 说某个时间段开放某明星演唱会订票，大概会同时有500K QPS的访问量，一共只有50K张票。 订票的过程是用户打开订票网页（不用考虑认证等问题），填一个text box说要订几张票，然后click一个button就打开一个page，那个page会不停的spin直到系统能够预留那几张票，如果预留成功，用户会有几分钟时间填写用户信息已经完成支付，如果到期未支付，这些票就自动被系统收回了。每张票都是一样的，没有位置信息什么的。

求教怎么design这个系统，我一开始看到500K QPS就有点慌乱了。

九章讲师解答
SOLUTIONS

订票网站一直都是世界难题。12306应该比这个还恐怖。
不要被 500k QPS吓到。500k也好，5k也好，500也好，分析的方法和思路都是一样的。

这个题的关键首先是给出一个可行解（无论任何系统设计题的关键都是如此），一个核心要实现的功能是，票的预留与回收。在设计可行解的时候，可以先将500k qps抛之脑后。假设现在只有10个用户来买票。优化的事情，放到Evolve的那一步。

按照我们的SNAKE分析法来：

Scenario 设计些啥:

- 用户提交订票请求
- 客户端等待订票
- 预留票，用户完成支付
- 票过期，回收票
- 限制一场演唱会的票数。

Needs 设计得多牛？

- 500k QPS，面试官已经给出
- 响应时间——是在用户点击的一瞬间就要完成预定么？不是，可以让用户等个几分钟。也就是说，500K的请求，可以在若干分钟内完成就好了。因此所谓的 500k QPS，并不是Average QPS，只是说峰值是 500k QPS。而你要做的事情并不是在1秒之内完成500k的预定，而是把确认你收到了购票申请就好了。

Application 应用与服务

- ReservationService —— 用户提交一个预定请求，查询自己的预定状态
- TicketService —— 系统帮一个预定完成预定，生成具体的票

Kilobyte 数据如何存储与访问

1. ReservationService —— 用户提交了一个订票申请之后，把一条预定的数据写到数据库里。所以需要一个Reservation的table。大概包含的columns有：

> id(primary_key)
> created_at(timestamp)
> concert_id(foreign key)
> user_id(foreign key)
> tickets_count(int)
> status(int)

简单的说就是谁在什么时刻预定了哪个演唱会，预定了几张，当前预定状态是什么（等待，成功，失败）。

2. TicketService —— 系统从数据库中按照顺序选出预定，完成预定，预定成功的，生成对应的Ticket。表结构如下：

> id (primary key)
> created_at (timestamp)
> user_id (fk)
> concert_id (fk)
> reservation_id (fk)
> status (int) // 是否退票之类的

另外，我们当然还需要一个Concert的table，主要记录总共有多少票：

> id (primary key)
> title (string)
> description (text)

> start_at (timestamp)
> tickets_amount (int)
> remain_tickets_amount (int)
> ...

总结一下具体的一个Work Solution 的流程如下：
1. 用户提交一个预定，ReservationService 收到预定，存在数据库里，status=pending
2. 用户提交预定之后，跳转到一个等待订票结果的界面，该界面每隔5-10秒钟像服务器发送一个请求查询当前的预定状态
3. TicketService是一个单独执行的程序，你可以认为是一个死循环，不断检查数据库里是否有pending状态的票，取出一批票，比如1k张，然后顺利处理，创建对应的Tickets，修改对应的Reservation的status。

Evolve

分析一下上述的每个操作在500k qps的情况下会发生什么，以及该如何解决。

1. 用户提交一个预定，ReservationService 收到预定，存在数据库里，status=pending

也就是说，在一秒钟之内，我们要同时处理500k的预定请求，首先web server一台肯定搞不定，需要增加到大概500台，每台web server一秒钟同时处理1k的请求还是可以的。数据库如果只有一台的话，也很难承受这样大的请求。并且SQL和NoSQL这种数据库处理这个问题也会非常吃力。可以选用Redis这种既是内存级访问速度，又可以做持久化的key-value数据库。并且Redis自带一个队列的功能，非常适合我们订票的这个模型。Redis的存取效率大概是每秒钟几十k，那么也就是我们要大概20台Redis应该就可以了。我们可以按照 user_id 作为 shard key，分配到各个redis上。

2. 用户提交预定之后，跳转到一个等待订票结果的界面，该界面每隔5-10秒钟像服务器发送一个请求查询当前的预定状态

使用了redis的队列之后，如何查询一个预定信息是否在队列里呢？方法是reservation的基本信息除了放到队列里，还需要同时继续存一份在redis里。队列里可以只放reservation_id。此时reservation_id可以用user_id+concert_id+timestamp来表示。

3. TicketService是一个单独执行的程序，你可以认为是一个死循环，不断检查数据库里是否有pending状态的票，取出一批票，比如1k张，然后顺利处理，创建对应的Tickets，修改对应的Reservation的status。

为每个Redis的数据库后面添加一个TicketService的程序（在某台机器上跑着），每个TicketService负责一个Redis数据库。该程序每次从Redis的队列中读出最多1k的数据，然后计算一下有需要多少张票，比如2k，然后访问Concert的数据库。问Concert要2k的票，如果还剩有那么多，那么就remain_tickets_amount - 2k，如果不够的话，就返回还有多少张票，并把remain_tickets_acount清零。这个过程要对数据库进行加锁，可以用数据库自己带的锁，也可以用zookeeper之类的分布式锁。因为现在是1k为一组进行处理，所以这个过程不会很慢，存Concert的数据库也不需要很多，一台就够了。因为就算是500k的话，分成500组，也就是500个queries峰值，数据库处理起来绰绰有余额。

假如得到了2k张票的额度之后，就顺序处理这1k个reservation，然后对每个reservation生成对应的tickets，并在redis中标记reservation的状态，这里的话，tickets的table大概就会产生2k条的insert，所以tickets的数据库需要大概能够承受 20 x 1k = 20k 的并发写。这个的话，大概 20 台SQL数据库也就搞定了。

从头理一下

开放订票，500k的请求从世界各地涌来
通过 Load Balancer 纷发给500台 Web Server 。每台Web Server大概一秒钟处理1k的请求
Web Server 将1k的请求，按照 user_id 进行 shard，丢给对应的 redis 服务器里的队列，并把Reservation 信息也丢给 Redis存储。
此时，20台 Redis，每台 Redis 约收到 25k 的 排队订票记录
每台 Redis 背后对应一个 TicketService 的程序，不断的查看 Redis 里的队列是否有订票记录，如果有的话，一次拿出1k个订票记录进行处理，问Concert 要额度，然后把1k的reservation对应的创建出2k左右的tickets出来（假如一个reservation有2张票平均）。假如这个部分的处理能力是1k/s的话，那么这个过程完成需要25秒。也就是说，对于用户来说，最慢大概25秒之后，就知道自己有没有订上票了，平均等待时间应该低于10秒，因为当concert的票卖完了的时候，就无需生成1-2k条新的tickets，那么这个时候速度会快很多。
存储tickets的数据库需要多台，因为需要处理的请求大概是20k的qps，所以大概20台左右的Ticket数据库。

超时的票回收

增加一个RecycleService。这个RecycleService 不断访问 Tickets 的数据库，看看有没有超时的票，如果超时了，那么就回收，并且去Concert的数据库里把remain_tickets_acount 增加。

总结如何攻破 500k QPS的核心点

核心点就是，500k QPS 我只要做到收，不需要做到处理，那么500台web服务器+20台Redis就可以了。
处理的的时候，分成1k一组进行处理，让用户多等个几秒钟，问题不大。用户等10秒钟的话，我们需要的服务器数目就降低10-20倍，这是个tradeoff，需要好好权衡的。

一些可能的疑惑和可以继续进化的地方

问：500台Web服务器很多，而且除了订票的那几秒种，大部分的时候都是闲置浪费的，怎么办？
答：用AWS的弹性计算服务，为每场演唱会的火爆指数进行评估，然后预先开好机器，用完之后就可以销毁掉。

问：为什么不直接用Redis也来存储所有的数据信息？

答：因为是针对通同一个Concert的预定，大家需要访问同一条数据（remain_tickets_acount)，shard是不管用的，Redis也承受不住500k QPS 对同一条数据进行读写，并且还要加锁之类的保证一致性。所以这个对 remain_tickets_acount 的值进行修改，创建对应的 tickets 的过程，是不能在用户请求的时候，实时完成的，需要延迟进行。

问：redis又用来做队列，又用来做Reservation 表的存储，是否有点乱？

答：是的，所以一个更好的办法是，只把redis当做队列来用 和 Reservation 信息的Cache来用。当一个Reservation 被处理的时候，再到SQL数据库里生成对应的持久化记录。这样的好处是，Redis这种结构其实不是很擅长做持久化数据的存储，我们一般都还是拿来当队列和cache用得比较多。

我用了2个小时来写这个帖子回复你，是不是该给我一个小红花！

阅读原文