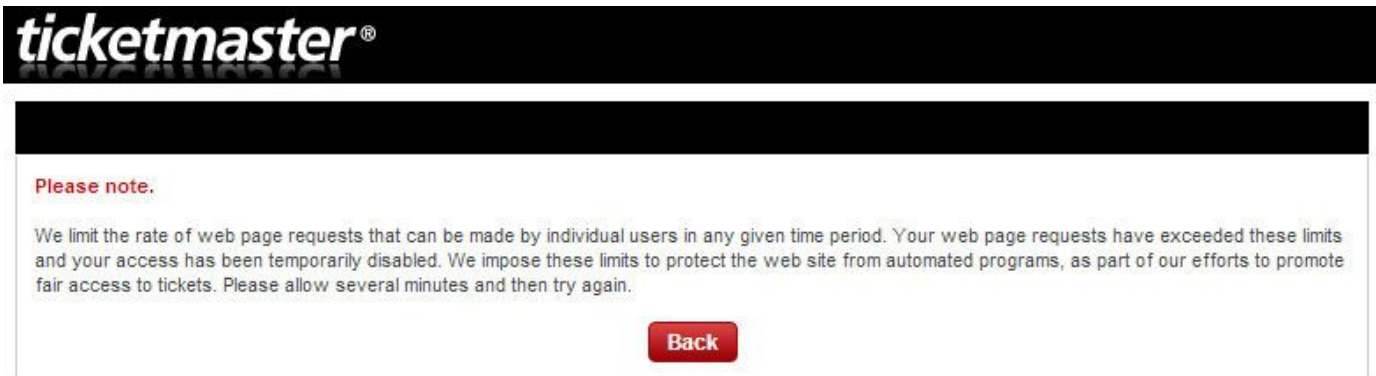



Interviewer: How to limit requests?

如何限制访问次数？

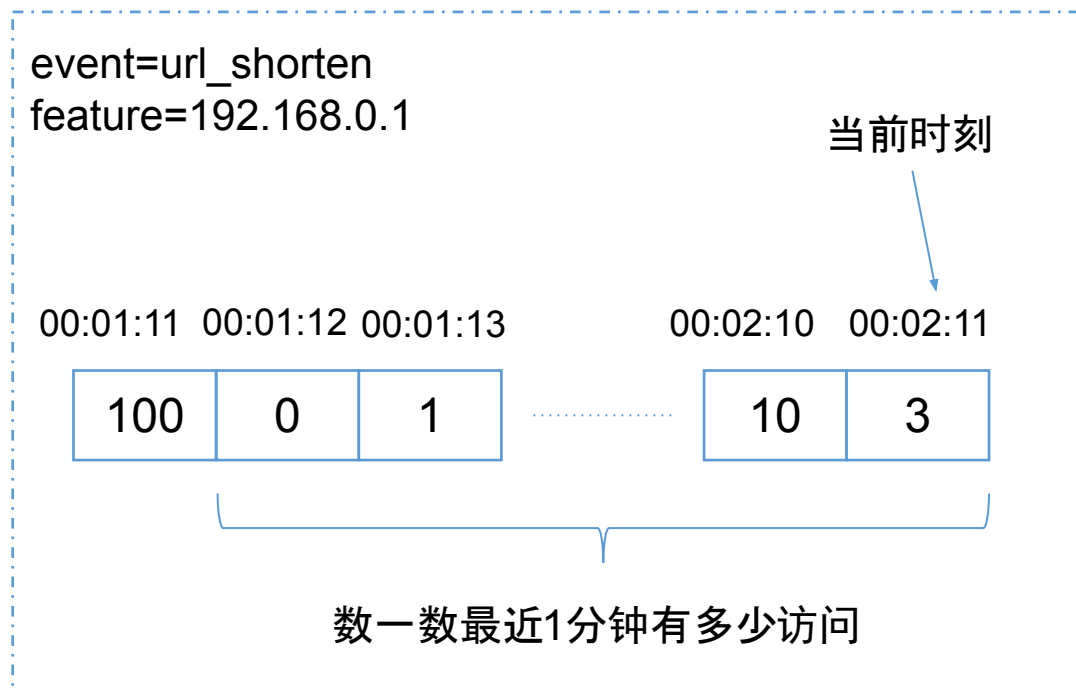
比如 1 小时之类不能重置 >5 次密码



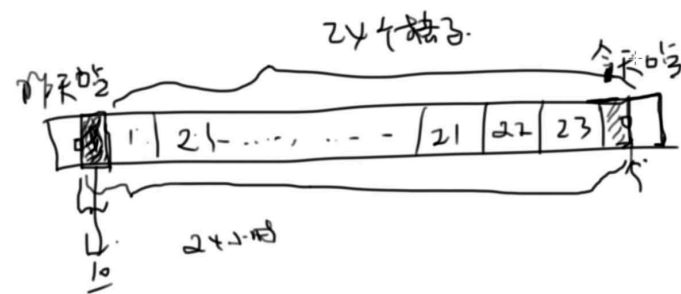
- Rate limiter
 - 网站必用工具
 - 比如一分钟来自同一个邮箱的密码输入错误不能超过5次, 一天不超过10次
- 一些Open source的资源
 - Ruby: <https://github.com/ejfinneran/ratelimit>
 - Django: <https://github.com/jsocol/django-ratelimit>
 - 建议: 有空读一读源码
- Rate Limiter 已经是一个小型的系统设计问题
- 我们同样可以用 4S 分析法进行分析!

- Scenario 场景
 - 根据网络请求的特征进行限制(feature的选取)
 - IP (未登录时的行为), User(登录后的行为), Email(注册, 登录, 激活)
 - 系统需要做到怎样的程度
 - 如果在某个时间段内超过了一定数目, 就拒绝该请求, 返回 4xx 错误
 - 2/s, 5/m, 10/h, 100/d
 - 无需做到最近30秒, 最近21分钟这样的限制。粒度太细意义不大
- Service服务
 - 本身就是一个最小的 Application 了, 无法再细分
- Storage 数据存取
 - 需要记录(log)某个特征(feature)在哪个时刻(time)做了什么事情(event)
 - 该数据信息最多保留一天(对于 rate=5/m 的限制, 那么一次log在一分钟以后已经没有存在的意义了)
 - 必须是可以高效存取的结构(本来就是为了限制对数据库的读写太多, 所以自己的效率必须高与数据库)
 - 所以使用 Memcached 作为存储结构(数据无需持久化) 

- 算法描述
- 用 event+feature+timestamp 作为 memcached 的key
- 记录一次访问:
 - 代码: `memcached.increament(key, ttl=60s)`
 - 解释: 将对应bucket的访问次数+1, 设置60秒后失效
- 查询是否超过限制
 - 代码:
 - for t in 0~59 do
 - `key = event+feature+(current_timestamp - t)`
 - `sum += memcached.get(key, default=0)` 
 - Check sum is in limitation
 - 解释: 把最近1分钟的访问记录加和



- 问:对于一天来说,有86400秒,检查一次就要 86k 的 cache 访问,如何优化?
- 答:分级存储。
 - 之前限制以1分钟为单位的时候,每个bucket的大小是1秒,一次查询最多60次读
 - 现在限制以1天为单位的时候,每个bucket以小时为单位存储,一次查询最多24次读
 - 同理如果以小时为单位,那么每个bucket设置为1分钟,一次查询最多60次读
- 问:上述的方法中存在误差,如何解决误差?
 - 首先这个误差其实不用解决,访问限制器不需要做到绝对精确。
 - 其次如果真要解决的话,可以将每次log的信息分别存入3级的bucket(秒,分钟,小时)
 - 在获得最近1天的访问次数时,比如当前时刻是23:30:33, 加总如下的几项:
 - 在秒的bucket里加和 23:30:00 ~ 23:30:33(计34次查询)
 - 在分的bucket里加和 23:00 ~ 23:29(计30次查询)
 - 在时的bucket里加和 00 ~ 22(计23次查询)
 - 在秒的bucket里加和昨天 23:30:34 ~ 23:30:59 (计26次查询)
 - 在分的bucket里加和昨天 23:31 ~ 23:59(计29次查询)
 - 总计耗费 $34 + 30 + 23 + 26 + 29 = 142$ 次cache查询, 可以接受



你觉得是否需要解决误差?

Interviewer: Design Datadog (Monitor System)

Google url shortener

<http://goo.gl/rN7W>

<http://www.sina.com.cn/>

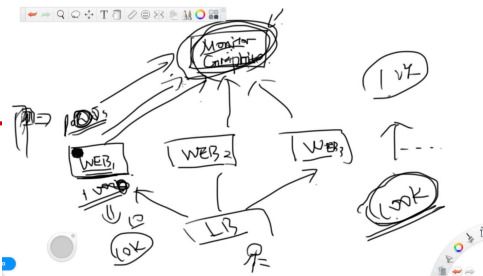
Created: 2009 Dec 16



怎样统计访问数据？

- <https://www.datadoghq.com/>
- 同样进行 4S 分析！
- Scenario 设计些啥
 - 对于用户对于某个链接的每次访问，记录为一次访问
 - 可以查询某个链接的被访问次数
 - 知道总共多少次访问
 - 知道最近的x小时/x天/x月/x年的访问曲线图
 - 假设 Tiny URL 的读请求约 2k 的QPS
- Service
 - 自身为一个独立的Application，无法更细分

怎样统计访问数据？



怎样统计访问数据？

- 问: 2k的QPS这么大, 往NoSQL的写入操作也这么多么?
- 答: 不是。
 - 可以先将最近15秒种的访问次数 Aggregate 到一起, 写在内存里
 - 每隔15秒将记录写进NoSQL一次, 这样写QPS就降到了100多
- 问: 如何得将昨天的数据按照5分钟的bucket进行整理?
- 答: 对老数据进行瘦身
 - 当读发现一个key的value比较多时, 就触发一次“瘦身”操作
 - 瘦身操作把所有老的记录进行 Aggregate
 - 这些旧数据的记录的专业名词叫做: Retention

Storage

- 基本全是写操作, 读操作很少
- 需要持久化存储(没memcached什么事了)
- SQL or NoSQL or File System?
 - 其实都可以, 业界的一些系统比如 Graphite 用的是文件系统存储
 - 这里我们假设用NoSQL存储吧
- 用NoSQL的话, key 就是 tiny url 的 short_key, value是这个key的所有访问记录的统计数据
 - 你可能会奇怪为什么value可以存下一个key的所有访问数据(比如1整年)
- 我们来看看value的结构
- 核心点是:
 - 今天的数据, 我们以分钟为单位存储
 - 昨天的数据, 可能就以5分钟为单位存储
 - 上个月的数据, 可能就以1小时为单位存储
 - 去年的数据, 就以周为单位存储
 - ...
 - 用户的查询操作通常是查询某个时刻到当前时刻的曲线图**
 - 也就意味着, 对于去年的数据, 你没有必要一分钟一分钟的进行保存**
- 多级Bucket的思路, 是不是和Rate Limiter如出一辙!

```
...
2016/02/26 23 1h 200
...
2016/03/27 23:50 5m 40
2016/03/27 23:55 5m 30
2016/03/28 00:00 1m 10
2016/03/28 00:01 1m 21
...
2016/03/28 16:00 m 2
```