伍正云：饿了么风控计数服务实战解析

作者/分享人：Chat 实录

查看本场Chat

向作者提问

2017年9月11日，周一晚上8点30分。2016年4月加入饿了么，目前在风控部门负责 faraday 计数服务、gaos 变量引擎、mike、tesla 等风控相关服务的伍正云带来了主题为《从0到1：饿了么风控计数服务是如何炼成的》的交流。以下是主持人哈比整理的问答实录，记录了作者和读者问答的精彩时刻。

内容提要：

- 中间值是历史的时间窗口计数器计算而来的结果。这句话的意思是记录每一次人为统计某时间段后保存下来的值，还是系统自动统计各种累加值之后的历史值存档?
- 假如 uesrid:888 的用户在 14:00-14:10 下单 5 次，14:10-15:00 下单 8 次，15:00-16:00 下单 3 次，16:00-17:10 下单 2 次。在 17:15 分时获取该用户最近两小时的下单数，值是多少？计算逻辑是什么?
- 饿了么除了这种计数器服务反作弊，有实时规则反作弊系统吗？实时规则和离线规则如何配合?
- 如果是集群部署，localqueue+getset 这种方式需要保证对应同一个 key 的请求落在同一台服务器上吗？ 如果不是，不同 server 之间对同一个 key 在各自的 localqueue 里的值都可能不

一样。高并发场景下，这个判断会大量失效吗？同样，在此流程中 getandset 能保证的是至少将本机的最新值写入，但这个值在集群范围内是否是最新的，就不一定了吗？

- 为了降低成本，faraday 的 key 去掉了一些不必要的 33 个 byte 的前缀，可以请举例说明哪些是属于不必要的吗？这个取舍需要和其他部门的人一起商讨吗，主要需要请教哪些人员的意见？
- 时间戳用 yyyyMMdd 字符串代替 timestamp 存储的优势是便于阅读，其次是节省内存，那这种方案有没有什么缺点？
- 正如文章中说到的，Redis 的单机性能很不错，但是它是纯内存的，所以成本较高。除了 Redis，饿了么还考虑过别的方案吗？它们相比 Redis 主要有什么优劣势？
- 如果并发请求高，计数服务与实践值偏差会很大。请问有弥补机制吗？
- 请问老系统更换 faraday 系统过程中有遇到什么困难吗？怎么处理的？
- 请问 mq 选用的技术方案是什么？ 消费异常时怎么做容灾？ 高并发下对热点 key 操作，redis 如何做容灾？如果 redis 不做持久化存储，如何做存储容灾？
- incr 有两种方式，一种是 long 一种是 double。如果是统计金额，long 肯定不实用，double 则会出现精度问题。请问这种情况怎么解决？
- 如果做了一个这样的活动，活动会记录参与活动的人数，要求每次有用户过来必须至少读一次和计数一次。但是活动太火爆，或者有异常流量过来，接入层拦截不掉：qps 超过了单台 redis 的读写极限。这种情况有什么好的处理方式吗？
- 使用 localqueue 的方式在集群环境下如何考虑？
- 能详细说下中间值吗？它是否等同于历史值，它需要多长时间计算或者通过什么规则计算？

问：中间值是历史的时间窗口计数器计算而来的结果。这句话的意思是记录每一次人为统计某时间段后保存下来的值，还是系统自动统计各种累加值之后的历史值存档？

答：中间值指的是历史计算下来的总值。

比如你计算最近 10 个小时的 count 值和第一次 get 值，你取的是 10 个小时的值，然后一个一个累加。同时你将其中过去 9 个小时累加成 count。当你第二次再次请求最近 10 个小时的 count，就去拿当前小时 get 值 + 过去计算好的中间值（也就是过去 9 个小时累加好的值）。

问：假如 uesrid:888 的用户在 14:00-14:10 下单 5 次，14:10-15:00 下单 8 次，15:00-16:00 下单 3 次，16:00-17:10 下单 2 次。在 17:15 分时获取该用户最近两小时的下单数，值是多少？计算逻辑是什么？

答：首先设计的时候不会出现小时和分钟混搭这种情况，redis 中组装 key 跟时间窗口挂钩。

比如你想统计最近 2 小时的计数器，当前时间是 14:00。xxx:2017091114 代表的是 14:00 的计数器，假设 10;xxx:2017091115 代表的是 15:00 的计数器，假设为 5；那么当前最近 2 小时的计数器就是用 14:00 加上 15:00，也就是 10+5=15。另外，如果你希望用更细粒度的计数器，就应该将时间窗口设置成分钟级别的滑动窗口。

问：饿了么除了这种计数器服务反作弊，有实时规则反作弊系统吗？实时规则和离线规则如何配合?

答：饿了么有规则引擎和模型引擎，两种在线规则的方式找出刷单用户和商户。规则引擎主要依赖计数器，模型引擎主要利用 spark 和 tensorflow 跑一层神经网络跑出商户和用户的作弊模型。

离线规则主要是通过 spark 和 tensorflow 跑多层神经网络。离线跑出来的嫌疑商户，通过脚本推送到 db，后台系统会读取 db 供运营人员判罚商户。

问：如果是集群部署，localqueue+getset 这种方式需要保证对应同一个 key 的请求落在同一台服务器上吗？ 如果不是，不同 server 之间对同一个 key 在各自的 localqueue 里的值都可能不一样。高并发场景下，这个判断会大量失效吗？同样，在此流程中 getandset 能保证的是至少将本机的最新值写入，但这个值在集群范围内是否是最新的，就不一定了吗?

答：首先是不保证同一个 key 的请求落在一台服务器上的。

有一点在文章细节中没说明白，第一是 local queue 取到值之后，会去调用 getset 方法拿到旧值，之后再去比较新值。如果有问题，会再去调用 getset 去做一次操作。第一次 getset 可以保证本机服务是没问题的，第二次 getset 可以保证大部分其他服务是没问题的。

这个设计在当初确实是一个折中方案。如果你希望强一致性，确实需要牺牲点性能。比如将所有的值入库，然后通过脚本定时轮训去比较。只不过这样系统就无法实时读取到最新的值。如果能容忍一两次的数据污染，通过 local queue 加上两次 getset，则确实是一个比较好的方案。

问：为了降低成本，faraday 的 key 去掉了一些不必要的 33 个 byte 的前缀，可以请举例说明哪些是属于不必要的吗？这个取舍需要和其他部门的人一起商讨吗，主要需要请教哪些人员的意见?

答：通常 redis key 的设计规范中，是需要加一些前缀用来做描述信息；在本次服务 key 的前缀。

counterId:001:mainbody:11811:time:20170911。如果将 counterId: mainbody:time 这些前缀去掉，可以为 redis 节省一点 key 的空间。因为计数器服务主要是针对本部门内的，所以不需要和其他部门的人讨论。

做这种改变主要是和架构师，以及本部门核心开发人员商讨。

问：时间戳用 yyyyMMdd 字符串代替 timestamp 存储的优势是便于阅读，其次是节省内存，那这种方案有没有什么缺点?

答：timestamp 指的是总秒数，这种设计 key 的长度太长，而且可读性不高；看到总秒数，你无法知道当前 key 的时间窗口，采用 yyyyMMdd 这种形式可以节省 key 的长度。目前对 yyyyMMdd 暂时还未发现有不妥之处，当然如果你设计的 key 对这种 timestamp 有其他用途也可以考虑，否则还是建议 yyyyMMdd 这种形式。

问：正如文章中说到的，Redis 的单机性能很不错，但是它是纯内存的，所以成本较高。除了 Redis，饿了么还考虑过别的方案吗，它们相比 Redis 主要有什么优劣势?

答：有考虑过公司正在自研发的 kvstore，但是 kvstore 有一个缺点就是没有大规模的在线使用，所以当初选用这个还是有一点担心。它的优点是持久化速度比 redis 高。还考虑过用纯 mysql，优点就是成本上比 redis 划算，缺点是效率上没有 redis 快。

所以，我们采用了 redis 做缓存，mysql 做持久化这种设计。当 redis 失效的时候，会从 mysql 读取计算，重新加载 redis 中。后期我们会考虑将天和月这种级别的时间窗口，直接通过离线加在线混搭模式计算。

问：如果并发请求高，计数服务与实践值偏差会很大。请问有弥补机制吗?

答：目前我们有一个离线的方案，是通过离线抽取 hive 计算历史的 max 和 min。如果两者不对等，则会以离线抽取的数值为准。在下一版本，我们会改造成将数据实时发往 kafaka，利用 spark 和 storm，定时计算这种 max 和 min 的数值，再将计算好后的数据推到 db 中，这样从 db 中取到的 max 和 min 的数值都是没有污染的。

问：请问老系统更换 faraday 系统过程中有遇到什么困难吗？怎么处理的?

答：第一是老系统的计数器如何迁移的问题，这种迁移成本还是比较高的。我们做法目前是通过双写的形式，新的计数器接入，老的计数器也保留。我们对计数器的计算时间只保留 3 个月，3 个月后，老计数器代码直接移除。

问：请问 mq 选用的技术方案是什么？ 消费异常时怎么做容灾？ 高并发下对热点 key 操作，redis 如何做容灾？如果 redis 不做持久化存储，如何做存储容灾?

答：外层的 mq，我们选用的是公司自己研发的 maxq，我们的 maxq可以推及消息2000万，maxq 有 ack 机制，如果出异常的话，它会重复发刚才那一条消息。local queue 我们使用的是 disruptor，主要看中了它的高性能。高并发下对热点 key 的操作，以及如何做容灾，是这样的，我们使用的是自己研发的 curors，我们对 redis 自己做了 sharding。持久化层是用的 mysql，当 redis 挂的时候，mysql 还可以继续提供服务。

问：incr 有两种方式，一种是 long 一种是 double。如果是统计金额，long 肯定不实用，double 则会出现精度问题。请问这种情况怎么解决?

答：如果对于精度要求比较高的这种情况，通常订单金额只保留 2 位小数，你可以 incr 的时候乘 100，当做整数存。取值的时候，再除 100。风控对金额要求不会像金融那么精确，我们只是拿这个数字当做标准，判断这个商户今天的 GMV 是否出现异常，不会有 0.0001 这种过高的要求。

问：如果做了一个这样的活动，活动会记录参与活动的人数，要求每次有用户过来必须至少读一次和计数一次。但是活动太火爆，或者有异常流量过来，接入层拦截不掉：qps 超过了单台 redis 的读写极限。这种情况有什么好的处理方式吗?

答：这种属于降级和限流方案了。

有两种方案：第一种是手动降级，也就是目前饿了么的做法。当系统出现异常时，通过打开降级开关，让系统恢复一段时间，之后运行正常。第二种利用 netflix 的开源框架 hystrix 框架，它可以设置错误率达到多少，降级几分钟，超时多少时间降级几分钟。

任何系统的设计，都无法保证百分百稳定正常，这也是 CAP 理论，不可能满足三者。既然 QPS 已经超过了单台 redis 的读写极限了，要么服务扩容 redis 做 sharding，要么降级，起码保证系统运行正常。

问：使用 localqueue 的方式在集群环境下如何考虑?

答：既然是 local queue 了，肯定是只针对本地队列了，它的设计是考虑到本地服务的一些特定情况。比如本地服务的入库，本地服务的数值比较等等。如果你觉得用多线程遇到瓶颈的话，不妨使用 local queue 加线程组合的方式。

问：能详细说下中间值吗？它是否等同于历史值，它需要多长时间计算或者通过什么规则计算?

答：目前我们的做法是采用懒加载的形式。如果没人调用这个计数器类型 A，那么计数器 A 是不会计算中间值的。只有调用了计数器 A，才会真正计算中间值。中间值和历史值相似，但不等同。比如你统计的最近 30 天，历史值是把 30 天的值累加计算好；而中间值是当前时间的统计值，加上过去 29 天的累加值。因为当前时间的值是变的。

写评论