统计方法

对统计浏览量有四个基本的要求：

* 计数必须达到实时或者接近实时。
* 每个用户在一个时间窗口内仅被记录一次。
* 帖子显示的统计数量的误差不能超过百分之几。
* 整个系统必须能在生产环境下，数秒内完成阅读计数的处理。

满足上面四个条件，其实比想象中要复杂。为了在实时统计的情况下保持精准度，我们需要知道某一个用户之前是否浏览过一篇文章，所以我们需要为每一篇文章存储浏览过它的用户的集合，并且在每次新增浏览时检查该集合进行去重复操作。

一个比较简单的解决方案是，为每篇文章维护一个哈希表，用文章ID作为key，去重的userid的集合(set数据结构)作为value。

这种方案在文章数量和阅读数比较小的情况下，还能很好的运行，但当数据量到达大规模时，它就不适用了。尤其是该文章变成了热门文章，阅读数迅速增长，有些受欢迎的文章的阅读者数量超过百万级别，想象一下维护一个超过百万的unqine userId的集合在内存中的，还有经受住不断的查询，集合中的用户是否存在。

自从我们决定不提供100%精准的数据后，我们开始考虑使用几种不同的基数估计算法。我们综合考虑下选出量两个可以满足需求的算法：

* 线性概率计算方法，它非常精确，但是需要的内存数量是根据用户数线性增长的。
* 基于HyperLogLog (HLL)的计算方法，HLL的内存增长是非线性的，但是统计的精准度和线性概率就不是同一级别的了。

为了更好的理解基于HLL的计算方法，究竟能够节省多少内存，我们这里使用一个例子。

考虑到r/pics文章，在本文开头提及，该文章收到了超过一百万用户的浏览过，如果我们存储一百万个唯一的用户ID，每一个id占用8个字节，那么仅仅一篇文章就需要8mb的空间存储！对照着HLL所需要的存储空间就非常少了，在这个例子中使用HLL计算方法仅需要 12kb的空间也就是第一种方法的0.15%。

(This article on High Scalability 这篇文章讲解了上面的两种算法.)

有很多的HLL实现是基于上面两种算法的结合而成的，也就是一开始统计数量少的情况下使用线性概率方法，当数量达到一定阈值时，切换为HLL方法。这种混合方法非常有用，不但能够为小量数据集提供精准性，也能为大量数据节省存储空间。该种实现方式的细节请参阅论文（Google’s HyperLogLog++ paper）

HLL算法的实现是相当标准的，这里有三种不同的实现方式，要注意的是，基于内存存储方案的HLL，这里我们只考虑Java和Scale两种实现

* Twitter的Algebird库，Scala实现，Algebird的文档撰写非常好，但是关于它是如何实现HLL的，不是很容易理解。
* stream-lib库中的HyperLogLog++实现，Java编写。stream-lib代码的文档化做的很好，但我们对如何适当调优它，还是有些困惑的。
* Redis的HLL实现(我们最终的选择)，我们觉得Redis的实现不管从文档完善程度还是配置和提供的API接口，来说做的都非常好。另外的加分点是，使用Redis可以减少我们对CPU和内存性能的担忧。



Reddit的数据管道，主要都是使用Apache Kafka的。每当一个用户浏览一篇文章时，就会触发一个事件并且被发送到事件收集服务器，然后批量的将这些事件发送打kafka中进行持久化。

Reddit的浏览统计系统，分为两个顺序执行的组成部分，其中的第一部分是，被称为Nazar的kafka队列『消费者』(consumer) ，它会从kafka中读取事件，然后将这些事件通过特定的条件进行过滤，判断改事件是否应该被算作一次文章阅读计数，它被称为『NAZAR』是因为在系统中它有作为『眼镜』的用处，识别出哪些事件是不应该被加入到统计中的。

Nazar使用Redis 维护状态还有一个事件不被计数的潜在原因，这个原因可能是用户短时间内重复浏览统一文章。Nazar会在事件被发送回kafka时，为事件添加一个标识位，根据该事件是否被加入到计数当中的布尔值。

统计系统的第二部是一个称为Abacus 的kafka『消费者』它会真正的统计浏览量，并且让浏览量数据可以在整站和客户端上显示， 它接收从Nazar发送出来的事件消息，然后根据该消息中包含着标识值（Nazar中处理的）来判断这个事件是否算做一次计数，如果事件被计数，Abacus会首先检查这个事件中文章的HLL计数是否存在于Redis中，如果存在，Abacus会发送一个PFADD请求给Redis，如果不存在，Abacus会发生一个请求到Cassandra集群，Cassandra集群会持久化HLL 计数和真实的原始计数数据，然后再发送一个SET请求到Redis，这个过程通常出现在用户阅读一个已经被Redis剔除的就文章的情况下发送。

为了让维护一个在Redis可能被剔除的旧文章，Abacus会定期的，从Redis中将HLL过滤数据，包括每篇文章的计数，全部写入到Cassandra集群中，当然为了避免集群过载，这个步骤会分为每篇文章10秒一组批次进行写入。下图就是整个过程的流程图。

