参考：<http://www.cnblogs.com/sunli/archive/2010/08/24/twitter_feeds_push_pull.html>

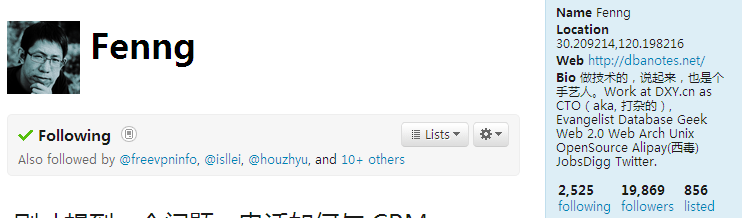
微博feed系统的推(push)模式和拉(pull)模式和时间分区拉模式架构探讨

     [文章作者：孙立 链接：<http://www.cnblogs.com/sunli/> 更新时间：2010-08-24]

     sns系统，微博系统都应用到了feed（每条微博或者sns里的新鲜事等我们称作feed）系统，不管是[twitter.com](http://twitter.com/)或者国内的新浪微博，人人网等，在各种技术社区，技术大会上都在分享自己的feed架构，也就是推拉模式（[timyang](http://timyang.net/)上次也分享了新浪微博的模式）。下面我们就微博的feed推拉(push,pull)模式做一下探讨，并提出新的时间分区拉模式。

      众所周知，在微博中，当你发表一篇微博，那么所有关注你的followers（粉丝）都会在一定的时间内收到你的微薄，这有点像群发一封邮件，所有的抄送者都会在一定的时间内收到。到这里，你可能觉得没有什么难度。我们看下下面的截图：



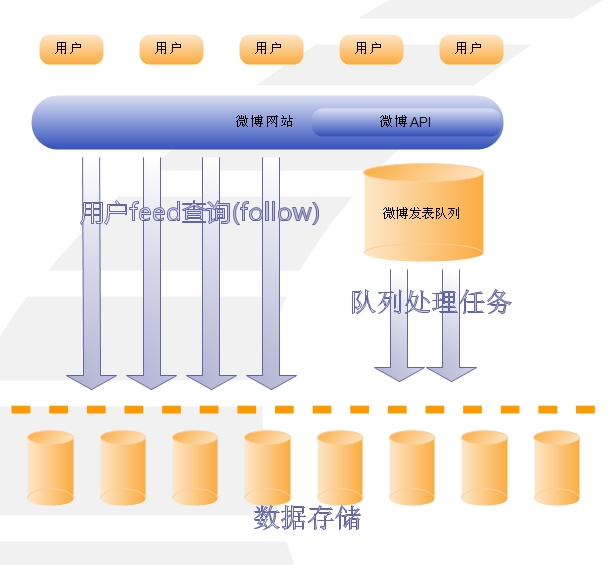
       图一：新浪微博姚晨  


         图二：twitter上冯大辉

     新浪微博的姚晨粉丝有2594751，她发表任何一篇微博，都需要2594751个粉丝在一定的时间内收到，twitter的冯大辉发表一篇的话，需要19868个followers收到。

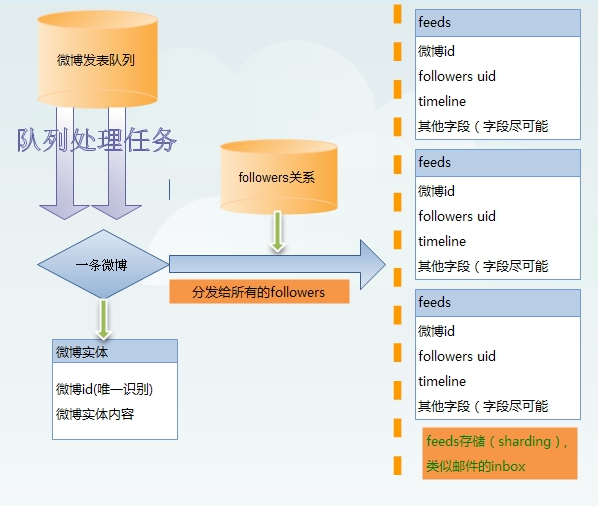
     相反，姚晨需要收到他关注的545个人的所有更新，冯大辉需要收到他关注的2525个人的所有更新。到这里，你是不是感觉到有那么一点点小挑战呢？

     下面我们看下微博一般的整体结构图：



                 图三：微博整体结构

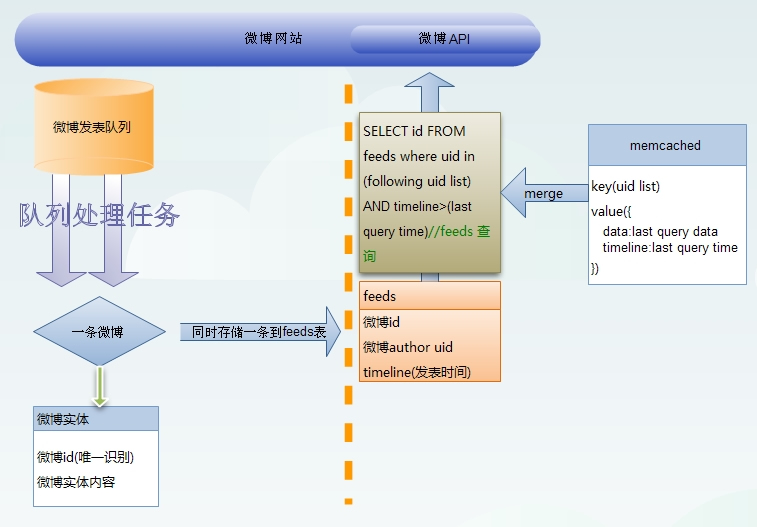
       图中展示了微博的整体数据流程，先了解下整体的数据结构，没有涉及到followers等的推拉模式处理。下面我们再看下推模式（push）：



          图四：推模式结构

   推模式需要把一篇微博推送给所有关注他的人（推给所有的粉丝），比如姚晨，我们就需要推送给2594751个用户的feeds表中。当然，feeds表可以很好的进行sharding,存储也都是一些数字型的字段，存储空间可能不是很大，用户在查询自己关注的所有人的feed时，速度快，性能非常高，但是推送量会非常大，姚晨发表一篇，就会产生200多万条数据。试想，一个大量用户的微薄系统通过使用推模式，是不是会产生非常惊人的数据呢？

    下面看下拉模式（pull）



            图五：拉模式

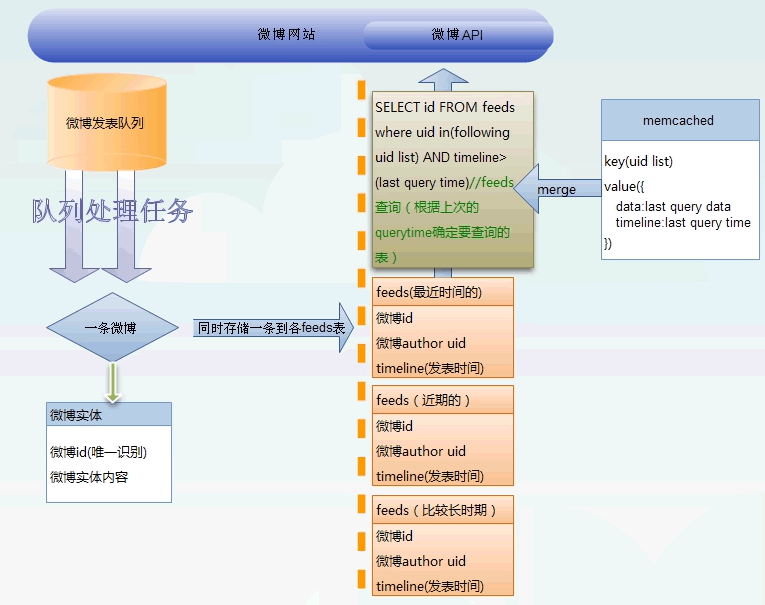
     拉模式只需要用户发表微博时，存储一条微博数据到feeds表中(feeds表可以是一个临时表，只保存近期可接受范围的数据).用户每次查询feed时都会去查询feeds表。比如姚晨打开自己的微薄首页，就产生：SELECT id FROM feeds where uid in(following uid list) ORDER BY id DESC LIMIT n(查询最新的n条)，缓存到memcached

uidlist=>{data:id list,timeline:上次查询出来的最新的一条数据的时间}

再次刷新：SELECT id FROM feeds where uid in(following uid list) AND timeline>(memcached存储的上次的timeline) ORDER BY id DESC LIMIT n

    这种模式实现起来也是比较简单和容易的，只是在查询的时候需要多考虑下缓存的结构。但是feeds表会产生很大的压力，怎么说feeds表也要保存最近十天半个月的数据吧，对于一个大点的系统，这会产生比较大的数据，如果following的人数比较多，数据库的压力就会非常大。而且一般在线的用户，客户端都会定期扫描，又会增加很大的压力，这在查询性能上没有推模式的效率高。

     下面我们在对拉模式做一下改进优化



           图五：拉模式(pull)-改进（时间分区拉模式）

      拉模式的改进主要是在feeds的存储上，使用按照时间进行分区存储。分为最近时间段（比如最近一个小时），近期的，比较长时期等等。我们再来看下查询的流程，比如姚晨登陆微博首页，假设缓存中没有任何数据，那么我们可以查询比较长时期的feeds表，然后进入缓存。下一次查询，通过查询缓存中的数据的timeline，如果timeline还在最近一个小时内，那么只需要查询最近一个小时的数据的feed表，最近一个小时的feeds表比图四的feeds表可要小很多，查询起来速度肯定快几个数量级了。

          改进模式的重点在于feeds的时间分区存储，根据上次查询的timeline来决定查询应该落在那个表。一般情况下，经常在线的用户，频繁使用的客户端扫描操作，经常登录的用户，都会落在最近的feeds表区间，查询都是比较高效的。只有那些十天，半个月才登录一次的用户需要去查询比较长时间的feeds大表，一旦查询过了，就又会落在最近时间区域，所以效率也是非常高的。

         关于时间的分区，需要根据数据量，用户访问特点进行一个合理的切分。如果数据发表量非常大，可以进行更多的分区。

        上面介绍的推模式和拉模式都有各自的特点，个人觉得时间分区拉模式弥补了图四的拉模式的很大的不足，是一个成本比较低廉的解决方案。当然，时间分区拉模式也可以结合推模式，根据某些特点来增加系统的性能。

        后记：本文的目的是介绍时间分区拉模式，本人对新浪微博和twitter等的推拉模式的细节并不清楚。

新浪微博：http://t.sina.com.cn/sunli1223