微博技术解密（上）| 微博信息流是如何实现的？

2019-01-29 胡忠想

从0开始学微服务 进入课程 >

讲述：胡忠想
时长 10:26 大小 9.57M

专栏结束后，有不少同学留言希望我能讲一些微博基础架构的知识。所以接下来的微博技术解密系列，我将分享微博在信息流架构、存储中间件等方面的经验，希望能给你带来启发和帮助。

今天我们先来看微博信息流架构，也就是微博的 Feed 是如何构建的。首先什么是 Feed 呢？根据我的理解，Feed 是互联网 2.0 时代的产物，它与互联网 1.0 时代的产物——门户网站最大的不同之处就是 Feed 不需要用户在各个板块之间来回跳转获取信息，而是把不同的信息都聚合在一起，可以供用户源源不断地访问。这里就涉及了两个问题，一个是信息如何保存，另一个是信息如何聚合。这也是今天我要分享的主要内容，我会从存储架构的角度阐述微博 Feed 是如何存储的，然后会从业务架构的角度阐述微博 Feed 是如何聚合的。

微博 Feed 存储架构

我们知道，微博 Feed 是由关注人的微博聚合在一起组成的，所以要存储每个人发的微博，那么在设计存储架构时主要需要注意三个问题：

每秒数据写入量，也就是每秒发博量是多大。

每秒数据访问量，也就是每秒微博请求量是多大。

是否有冷热数据之分，也就是微博的请求是否有时间特点。

结合微博的业务场景，我来回答上面提出的三个问题。首先是每秒发博量，这里要考虑到极端情况，比如元旦零点，瞬间会有大量用户发博，达到数万 QPS。再来看下每秒微博请求量，同样要考虑到在热点事件时，比如“春晚”时会有大量用户访问微博，请求量也会达到数万 QPS；并且每个用户关注的不止是一个人，假设关注数的平均值是 200，那么微博数据的请求量就是几百万 QPS。除此之外，微博的访问也是有时间特点的，用户一般访问新发微博的概率要远远大于一周前发的微博，所以说微博数据也是有冷热之分的。

这三个问题共同决定了微博的存储架构应该如何设计。在讨论微博存储架构前，我们先来看看目前业界比较成熟的存储方案，主要分为下面几种。

以 MySQL 为代表的关系型数据库。主要用来存储结构比较固定的数据，因为使用的是磁盘存储，所以写入和访问能力主要取决于磁盘的读写能力。而磁盘主要分为 SAS 盘和 SSD 盘，也就是机械盘和固态盘，两者的读写能力有一定的差距，SSD 盘读写能力是 SAS 盘的 3 倍左右，不过 QPS 都在千级别。磁盘存储的特点是不易丢失数据，可以永久保存。

以 Memcached 和 Redis 为代表的内存存储。服务器的内存大小一般要远小于磁盘，在几十 GB 到几百 GB 之间，而磁盘通常都是 TB 级。内存存储的优势就是读写速度快，读写能力能到几十万 QPS，远远大于磁盘存储。但由于数据存储在内存中，如果进程挂掉或者机器重启，内存中的数据就清空了。

HBase 为代表的分布式存储。属于非关系型数据库，它的特点是数据结构不固定，因此适合非结构化的数据存储，而且由于采用了分布式存储，使用 HDFS 作为底层文件存储系统，所以可以存储海量数据，并且具备非常高的写入性能。

讲到这里，结合前面提到的微博业务场景，你觉得存储架构该如何设计呢？

根据微博的实际业务情况，用户的微博需要永久保存，也就是进行持久化存储，而且微博的数据结构是固定的，优先考虑采用关系型数据库，因此 MySQL 是一种选择。但是考虑到

单台 MySQL 读能力的极限是不到一万 QPS，而微博的数据请求量是几百万 QPS，如果单纯使用 MySQL 来应对的话，需要上千台服务器，成本非常高。还有一点就是微博数据的请求是有冷热之分的，一周外的数据访问的概率要远小于一周内的数据。综合以上几点考虑，可以在 MySQL 存储的前面，再加一层缓存，比如使用 Memcached，存储最近一周内的微博，而用户的全量微博数据则持久化存储在 MySQL 中。假设用户访问一周内微博的概率是 99%，那么对缓存的请求量就是几百万 QPS，而对数据库的请求量就只有几万 QPS 了，而缓存的读写能力是几十万 QPS，相比较而言，需要的机器数要远小于只使用数据库了。

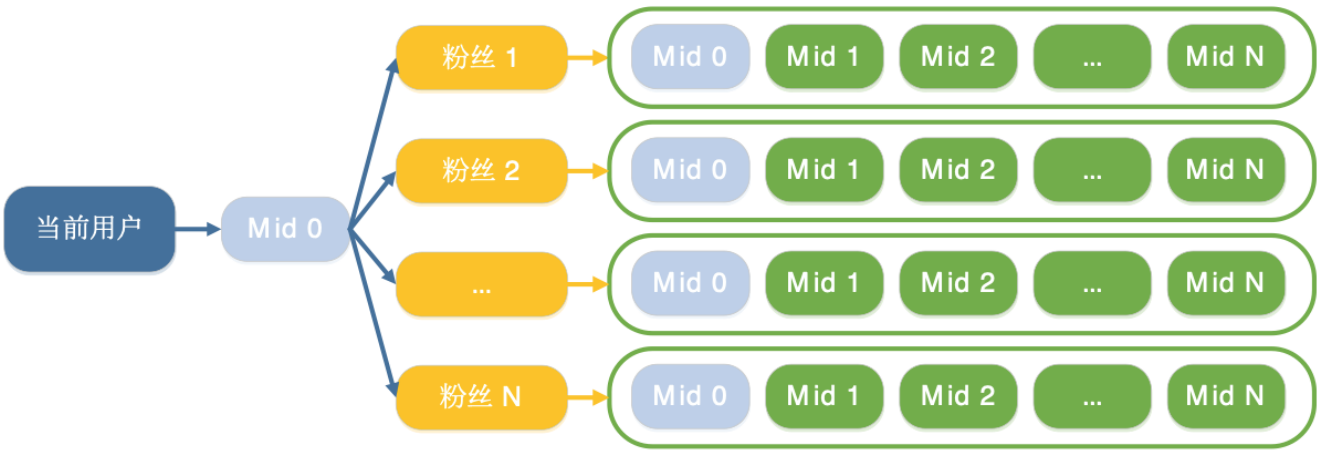
微博 Feed 业务架构

经过前面的讲解，假设我们已经使用 MySQL + Memcached 的双层存储架构解决了微博的存储问题，接下来面临的问题就是，如何将存储的关注人的微博数据聚合成一股源源不断的信息流以供用户访问。

根据我的经验，信息流聚合一般有三种架构：**推模式**、**拉模式**以及**推拉结合**，下面我来详细讲解这三种架构。

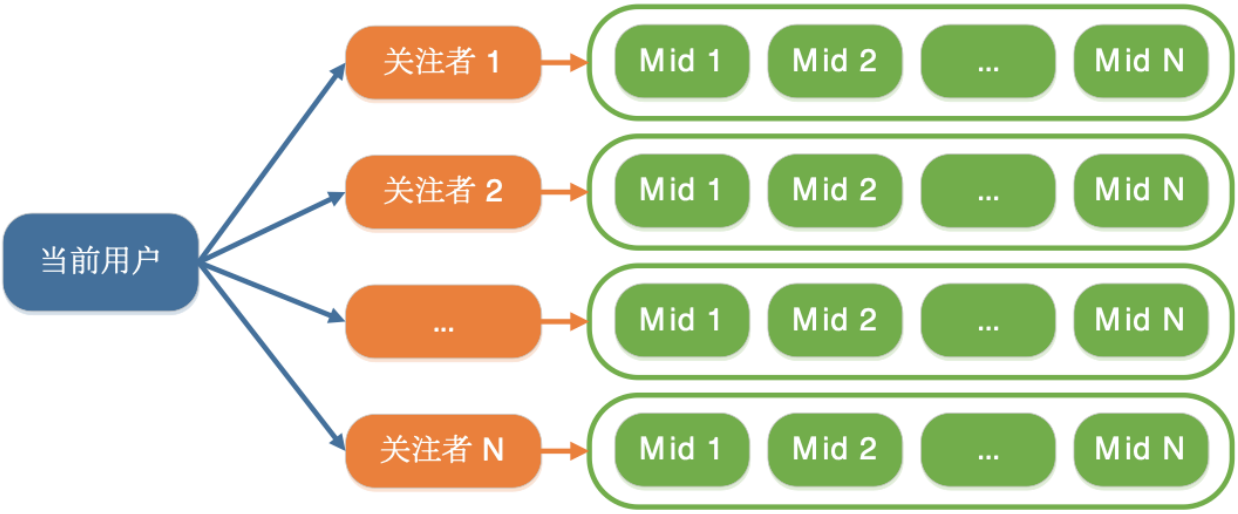
1. 推模式

推模式，顾名思义就是把关注人的发的微博，主动推送给粉丝，就像下图描述的那样。推模式相当于有一个收件箱，当关注人发微博的时候，就给所有粉丝的收件箱推送这条微博，这样的话，不管你关注了多少人，他们发的微博都会主动推送到你的收件箱，你只需要访问自己的收件箱，就可以获取到所有关注人发的微博了。



2. 拉模式

拉模式与推模式恰恰相反，就像图里那样每个人都有一个发件箱，发微博的时候就把微博存储到发件箱，这样的话，如果要获取所有关注人发的微博，就需要遍历关注人的发件箱列表，取出所有关注人的发件箱，然后按照时间顺序聚合在一起。



刷微博　　发件箱

我们先来对比下微博 Feed 采用哪种架构比较合适。首先来看推模式，它的特点是聚合简单，每个人只要查看自己的收件箱就可以获取到关注人发的所有微博，但缺点是即使粉丝没有请求 Feed，每个人发的微博也都要给所有粉丝推送一遍，所以写入量会非常高，尤其是对于具有上亿粉丝的用户来说，发一条微博需要给上亿粉丝的收件箱推送这条微博，这个写入量是非常大的，给处理机和数据库带来的压力可想而知。并且同一条微博会存储多份，占用的存储空间也非常大，并且如果需要修改或者删除微博，需要请求所有粉丝的收件箱。考虑到微博是公开的社交媒体平台，拥有上千万粉丝的用户不在少数，所以采用推模式的话，存储成本以及数据更新成本会非常高，需要大量的缓存和数据库以及队列机来更新。早期 Twitter 的 Feed 就采用了推模式，经常出现更新延迟，就是因为大量的写入带来巨大压力所导致。

再来看下拉模式，它的特点是聚合逻辑复杂，每个人要想查看关注人发的所有微博，需要遍历关注人列表，获取所有微博后再按照时间聚合在一起，对数据的请求量和聚合带来的计算量要远远大于推模式。但因为每个用户的微博只存在自己的发件箱列表中，所以相比于推模式来说，存储成本要小得多，并且如果有数据修改，只需要修改自己的发件箱列表就可以了。

3. 推拉结合

在对比了推模式和拉模式各自的优缺点之后，也就自然而然产生一种新想法，是不是可以采用推拉结合的方式，各取两者的优点呢？针对关注的粉丝量大的用户采用拉模式，而对于一般用户来说，他们的粉丝量有限，采用推模式问题不大，这样的话一个用户要获取所有关注人的微博，一方面要请求粉丝量大的关注人的发件箱列表，另一方面要请求自己的收件箱列表，再把两者聚合在一起就可以得到完整的 Feed 了。

虽然推拉结合的方式看似更加合理，但是由此带来的业务复杂度就比较高了，因为用户的粉丝数是不断变化的，所以对于哪些用户使用推模式，哪些用户使用拉模式，维护起来成本就很高了。所以综合考量下来，微博 Feed 采用了拉模式。

前面提到采用拉模式的话，需要拉取所有关注人的发件箱，在关注人只有几十几百个的时候，获取效率还是非常高的。但是当关注人上千以后，耗时就会增加很多，实际验证获取超过 4000 个用户的发件箱，耗时要几百 ms，并且长尾请求（也就是单次请求耗时超过 1s）的概率也会大大增加。为了解决关注人数上千的用户拉取 Feed 效率低的问题，我们采用了分而治之的思想，在拉取之前把用户的关注人分为几组，并行拉取，这样的话就把一次性的聚合计算操作给分解成多次聚合计算操作，最后再把多次聚合计算操作的结果汇总在一起，类似于 MapReduce 的思路。经过我们的实际验证，通过这种方法可以有效地降级关注人数上千用户拉取 Feed 的耗时，长尾请求的数量也大大减少了。

总结

今天我给你讲解了微博 Feed 的存储架构以及业务架构的选型，你可以看到最终方案都是结合微博的业务场景以及当时存储的成熟度做出的选择。微博诞生于 2009 年，并在 2010 年进行了平台化改造，确定了现在这一套存储和业务架构。当时的存储成熟度决定了 MySQL + Memcached 的组合是最优解，并没有使用 Redis、HBase 等后来更为先进的数据库；同时由于微博 Feed 的业务特点，选择了拉模式的业务架构，并针对关注人数上千的场景进行了拉取方式的优化，也被实践证明是行之有效的手段。

思考题

如果让你来设计微博的存储架构，除了使用 MySQL + Memcached，你觉得还可以使用哪些存储？

欢迎你在留言区写下自己的思考，与我一起讨论。

上一篇 阿忠伯的特别放送 | 答疑解惑02

下一篇 微博技术解密（下）| 微博存储的那些事儿

精选留言 (5)

写留言

平头哥
2019-02-15
2

存储在redis中,然后对微博信息进行聚合操作比较简单吧

Geek_59b08...
2019-05-05

目前，我信息流处理是Elastic search +redis+MySQL

展开

北极的大企...
2019-04-11

比较想问的是学完框架后，是先学设计模式还是先学JVM原理，并发与线程安全，然后中间件和架构设计，数据库设计，Linux学习，跨语言学习顺序，这些都是按照什么样的顺序学习的，还有源码阅读顺序

kext

2019-02-16

写入是tps，查询是qps

展开

苹果xixi

2019-01-30

1. memecached 维护问题 2. memecached 故障你们是如何解决的？