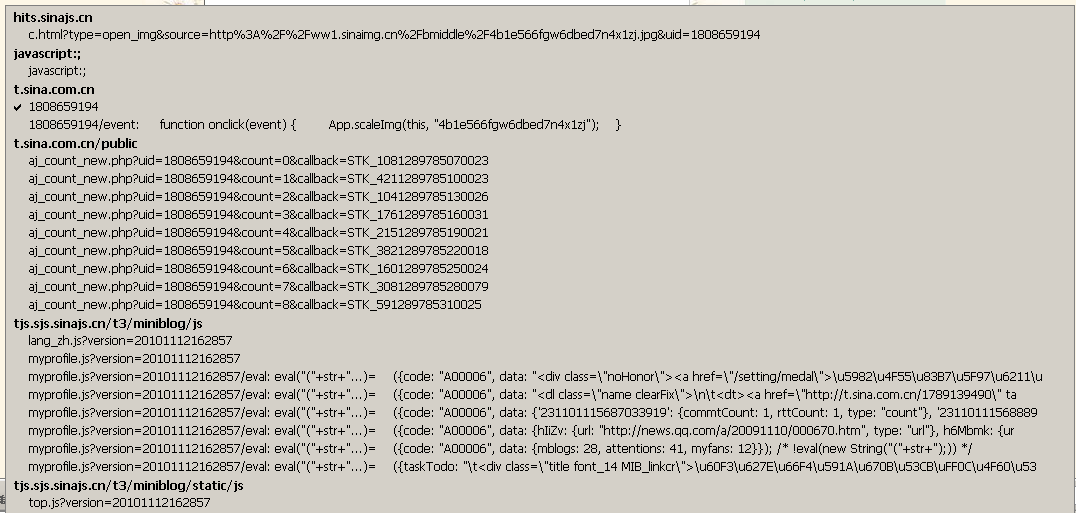
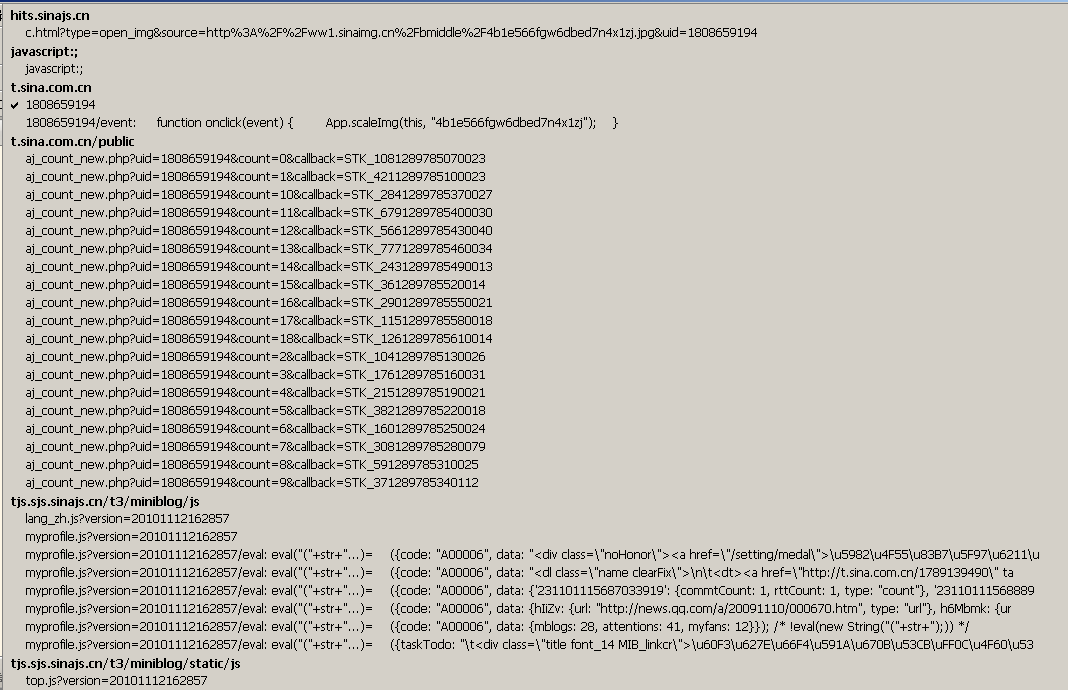
# 新浪微博架构猜想

定时轮询服务端，获得

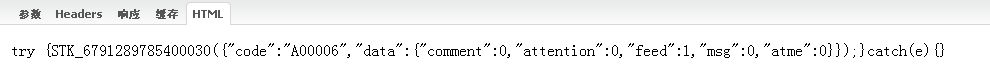


几分钟之后：

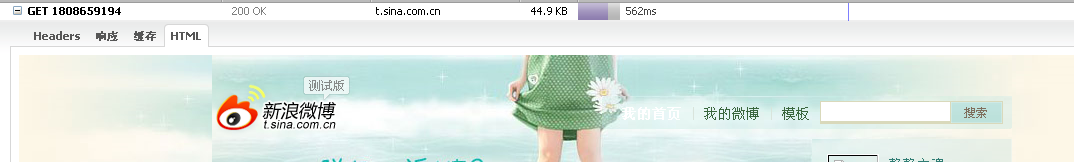


采用Jsonp的方式

STK\_\*\*\*\*\*这些js中的callback的方法是动态生成的



## 进入页面：



直接获得html代码，应该是采用了全页面缓存

这里面缓存的是页面的feed，不缓存好友数，头像这些，这些还是通过XHR来获得的

动态页面部分静态化

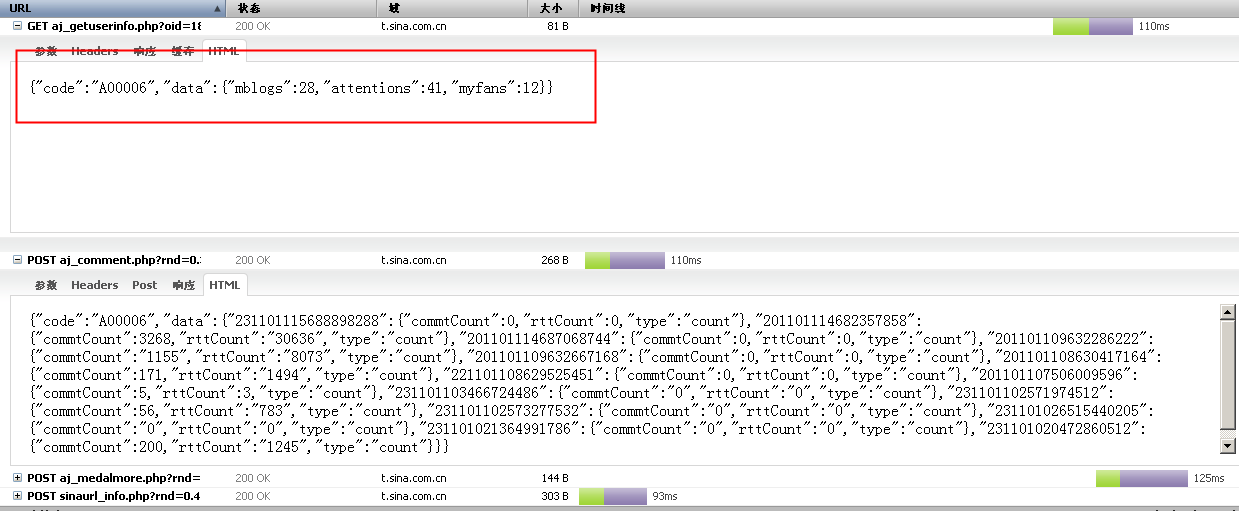
## 点击微博：



出现的也是通过content cache过的



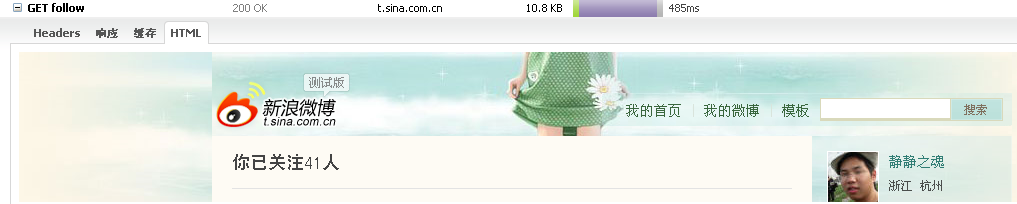
这里面没有显示粉丝数，关注数，以及每个feed的评论、转载数，根据新浪微博架构的描述，这些是动态变化的，如果这个也缓存，用户体验以及准确度都达不到要求，老是变动会失去缓存的意义，所以，这些是通过XHR去服务端请求的，如下图：



## 点击粉丝：



## 点击关注

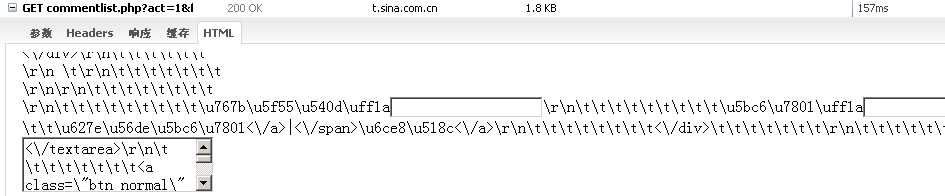


采用的技术和点击微博一样，全页面缓存

## PS：为什么新浪微博更新（新发表，评论，好友数等），需要手工去点击？

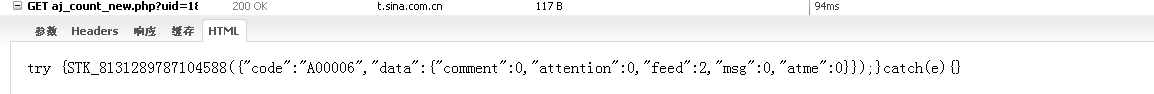
因为新浪微博采用了全页面缓存的方式，点击的动作刚好触发去做整个页面缓存。

## 评论：



直接返回的是评论内容的HTML，避免页面渲染

## 轮询feed



轮询回来的结果通过STK这个方法，来判断是否有更新，如果都为0，则不更新，否则提示有新的更新，然后让用户点击。

返回的是是否有新的评论，新的feed数，可见服务端肯定保存了我当前获得的那条最新记录数，应该是放在vector cache（feed id list）中，

## 架构：

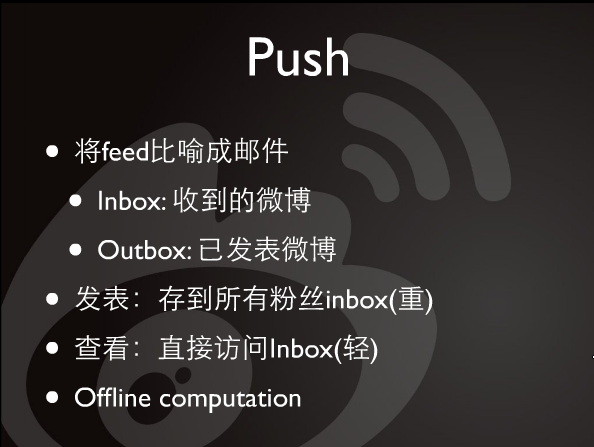
To Push or to pull It’S a question

新浪采用了什么技术来实现这个最核心的问题的呢？

### Feed架构

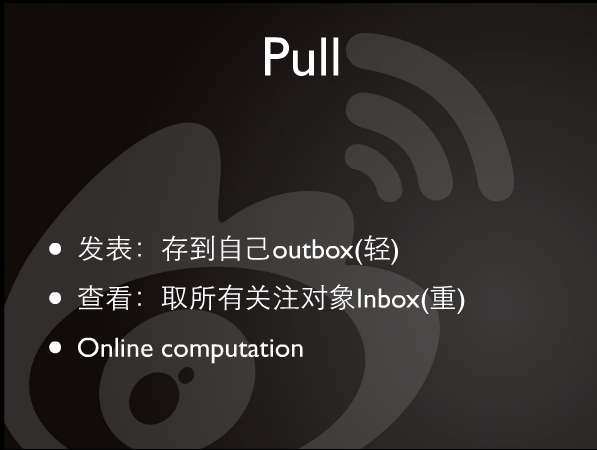


Push：



简单，但是当量打的时候，则分发到所有粉丝那边就变得比较麻烦，如小潘同学这种微博控，粉丝数有250w+，他发一条消息，如果采用push，简直就是自虐嘛

Pull：



存储量少，但是要在线进行计算，比如我有40个关注的对象，那我每次需要把40个人的feed都拿出来，然后根据时间进行排序，而且翻页也有问题，所以基本上纯pull不太靠谱

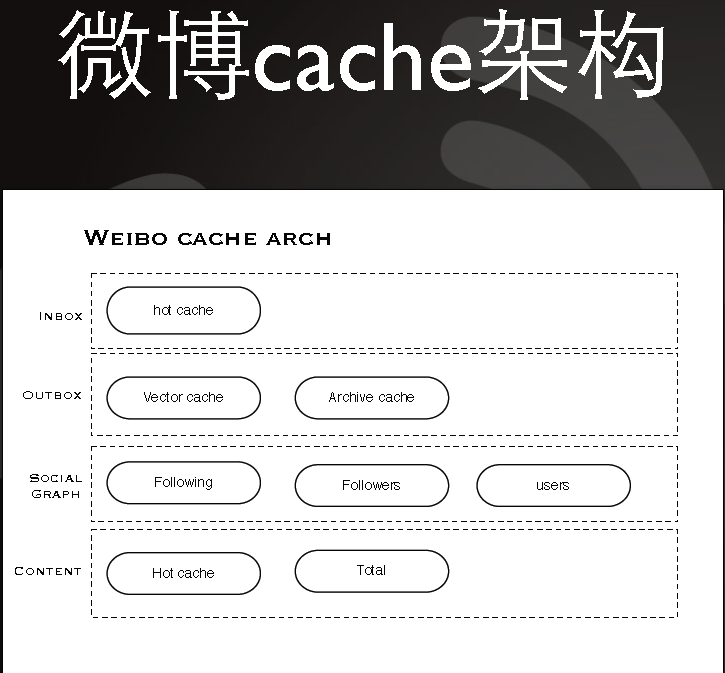
#### 新浪应该是采用复合型的方式，push+pull

Read：首先读inbox，然后在获取关注列表，然后读取关注列表中的outbox，最后信息进行聚合，生成新的inbox的vector cache

当第一次的时候，inbox的hot cache为空，则从所有关注列表的outbox的vector cache中获得id，然后进行聚合，最后生成一个新的inbox 的hot cache

Write：每个在线用户生成一个inbox，然后当关注的人发表一条微博的时候，会向这个人的inbox中插入一个id，格式如：feed{1,2,3,4…100}，这样可以减少所有用户那边插入数据，又可以减少计算的代价

### Cache



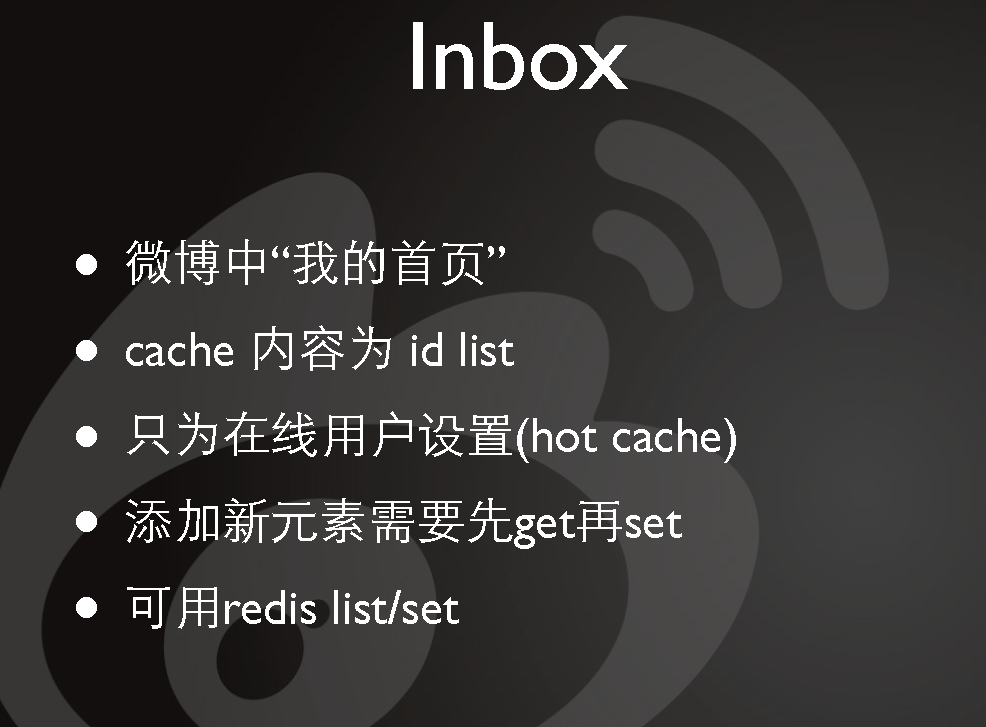
前面的Inbox和outBox和索引比较类似，只存储了feed的id列表

Social graph这个就是列表和用户的资料

最后一个是内容缓存，比如热门的那些内容放到hotCache中，还有就是全页面缓存以及预热生成json和xml给openAPI

#### Cache 结构

第一部分就是inbox：微博我的首页开始的ID列表，完全是内存里面，但是有一个缺点，我们要添加元素需要先GET再SET



第二部分outbox发出微博有存储最新ID在于聚合（key为自己）。为了高效，我们通常分两部分，第一就是说最新的一个ID列表比如说我们有100条内容，这个用户很久没有来，这个是空要过来取就要从我工作列表用ID地址聚合起来



第三部分是关注列表，这些都是纯ID，然后following，following加载开销比较大，上百万粉丝，越大的集合越容易变更，改变则需要delete all减少使用following list场景

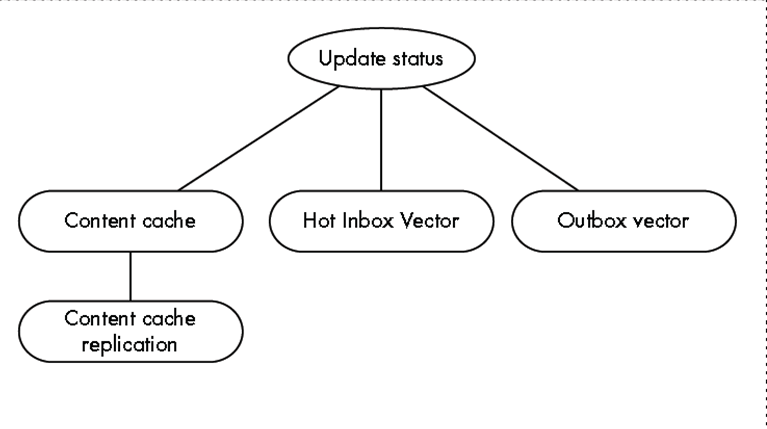


第四部分content cache微博内容体里面有一个很重要的内容，热内容。这个用户有200万粉丝，这个内容是热内容，在线粉丝也非常多，多分防止单点访问瓶颈，最终格式预生成，openAPI需要返回xml，json格式



#### Cache 流程

##### 发表一条微博



发表一条微博，会改变3个地方，

第一个是content cache，就是内容提缓存需要

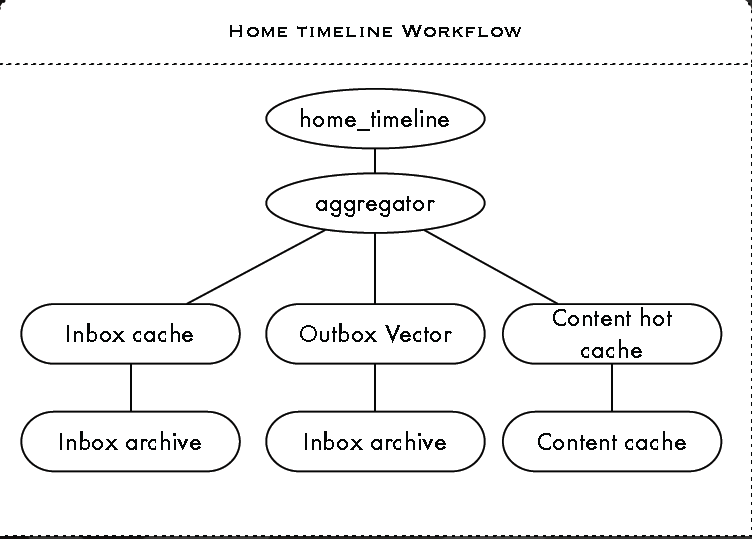
增加，然后进行复制，

第二个修改outbox，把自己发表的那个feed的id加到自己的outbox的vectory cache中

第三个就是根据粉丝列表修改inbox数据列表，如果自己的粉丝在线，则在这些粉丝的inbox中，插入一条这个feed的id这个过程是异步的

##### 获取首页Feed的流程

首先看inbox cache是否可用，获取关注列表，聚合内容从following关系，根据id list返回最终Feed聚合内容。最常用两个流程就是这样



##### 轮询

轮询自己的inbox，因为当需要轮询的时候，自己的inbox的hot cache已经有数据了，所以当关注的人发表一下，会想这个hotcache中插入一个id，这样轮询的时候就可以知道是否有更新

想法：

其实新浪为什么不这么做：

用户数少的时候，直接插入所有人的inbox，粉丝多的时候，不再插入他们的inbox，可以通过这样的方案来做，首先访问自己的inbox，如果有变更，则提醒，比如每3秒轮询自己的inbox，当第3次的时候，获取关注列表中那些粉丝数比较大的人，然后轮询他们这段时间内的outbox，然后在计算一把，这个实现方案是最简单的，但是新浪为什么不这么做？

实时性以及峰值计算的时候存在瓶颈