1. **总体架构**



下载器由三部分组成，Generator产生任务，Manager用于分配任务，Worker用于执行任务。新浪微博的数据可以通过两种通道获取，API和页面，任务产生时候要指定数据获取方式。

Generator产生任务的源可以来自各个地方，只要将相应任务放入Redis任务队列中即可。

需要下载的数据有两种，微博数据（评论、转发、收藏的更新，原始微博、转发微博的下载等）和账号信息（账号基本信息、粉丝、关注等）。

微博数据又可以分为两种，一是通过搜索抓取指定关键字相应的微博，二是抓取指定账号的微博。

搜索关键字总的抓取策略是通过网页抓（付费接口不够用），抓取任务只抓第一页，如果抓取的这一页微博时间晚于已经抓取的微博最新时间，中间差的微博数使用付费API接口获取。Manager根据经验值调整每个任务的抓取频率，达到抓取微博数据同时规避被封的目的。

抓取指定账号的微博优先通过API抓取（免费）。通过网页抓取策略同抓关键字。

微博账号数据的抓取优先通过API抓取，原因是由于用户隐私设置，网页抓取的数据大部分是不全的，只能获得少量基本信息。

Manager与Worker的通讯可以选择的有两种：Mina，Hessian，Mina是TCP协议，用于双向通讯，Hessian是HTTP协议，单向通讯。如果需要Manager主动向Worker推送任务，则选择Mina。Hessian的优点是比较轻量级。在效率上，内网中二者差别不是很大。

1. **功能**
   1. Generator

Generator中有各种任务的生成器，每种任务对应自己的任务队列，任务队列存储在Redis中；Generator也可以是其他任何程序，只要产生任务格式符合要求即可。

产生任务的方式有两种，一种是定时产生，一种是产生间隔时间需要变化调整。适用于第一种方式的一般是适用API进行抓取的，适用第二种的是使用网页抓取的任务。

每种任务队列需要设定自己的任务长度，如果任务队列满意味着任务处理能力不足，则任务插入需要延迟，同时做相应的报警。

* 1. Manager

Manager等待worker请求任务，如果有相应的任务则把任务分配出去，同时将任务放入已分配队列中，每个任务有唯一的ID，worker工作完成后，根据ID告知Manager任务执行情况，失败的任务根据情况是重新执行还是丢弃。

* 1. 网页抓取任务产生算法：根据历史经验值，计算任务产生频率。

设时间返回条微博，那么

其中 是一个常量，用于修正误差。这个公式的意思是，如果取回的微博数量超过时候取回的数量，那么就加快任务产生的频率，反之减小任务产生的频率，加上修正值是为了使产生任务的频率频繁，以尽量使用页面抓取。

* 1. 微博更新

微博数据抓下来，部分需要更新评论、转发数，用于发现热点微博，更新策略如下：

需要更新的微博存储到更新列表（可以存mongodb中），对第一次刚刚抓下来的微博全部更新，对已经更新的微博计算更新系数，更新系数随时间递减，初始化为负数，当递减为0的时候不予更新，更新列表定时清除不再更新的微博。

对于挖坟产生的历史微博，可以认为如果是热门的微博必定会抓到；另外监控账号的微博需要全部更新，但是频率可以比较低。

* 1. 微博传播路径

重点监控某一条微博的传播路径，需要将此微博的所有转发微博抓取下来，然后根据转发微博计算

* 1. Worker

为了避免被封Woker主要是模拟个人行为按照关键字去搜索，可以把Worker想象成一个工作人员，它取得一个任务后，按照任务搜索第一页，将数据拿到后，停留随机时间，然后再去取下一个任务。同一个IP不能并发的执行多个任务，可以通过增加代理IP数量不断的去取页面。

数据的存储由其他的模块提供接口供Worker调用。

每一个Worker拥有自己的IP和账号，同一个账号在同一个IP上面登录，IP和账号是绑定的，可以认为是同一个人一直在同一个IP上面进行操作，如果一旦出现IP被封的情况，马上停止使用此IP并报警；同时告诉自己的监控系统去检测此IP是否已经解封，如果解封了可以继续重新启动一个线程使用此IP。

如果Woker抓到的微博最老时间晚于库中已有的最新时间，那么中间差的微博使用api接口获取。

Worker中也可以单独起只使用API抓取的任务，此种任务只是更新代理，不会涉及注册的账号。

* 1. 微博账号的更新

有些账号需要不断更新粉丝关注以及账号本身的信息，可以通过API进行更新。

1. **存储**
   1. 抓取微博的存储

抓取的微博存入Hbase中，同时插入到更新表（mongodb），更新表中不包括微博的内容，只包括微博ID、转发数、评论数、作者ID、是否原创、平台信息、发布时间、插入时间、更新时间、所属搜索条件等信息。

* 1. 账号存储

存入mongodb中

* 1. 账号关系存储

存入mongodb。

* 1. 微博转发存储

存放微博ID与转发微博ID的队列对应关系，计划存入mongodb中。

1. **反扒策略（稍微总结一下）**

大多的信息可以通过免费API获得，主要是一些需要通过页面搜索的功能面临被禁的问题。

账号被封主要是指账号在页面操作时候出现要求输入验证码或绑定手机号激活的情况，主要原因如下：

1. 一个账号在非常规ip经常出现，例如以前曾经使用过国外的ip资源，其上微博账号出验证码/手机激活的概率要比国内大得多
2. 微博账号写操作频繁容易出验证码/手机激活，最早做微博账号自动运营时，每天定时会发若干微博，被搞的概率比读操作要高得多；甚至出现永久被封的状况
3. 在做微博搜索时，不恰当的操作（例如读得太频繁、ip使用不当，但并没有定量的标准）容易导致账号出问题
4. 如果一个ip出现的异常次数太多，ip会永久被封。

API的操作主要通过IP的操作次数来限制，这个可以参考新浪文档，一般不会永久封掉，只会在一段时间内限制访问。

现在反扒的主要策略是绑定账号和IP，尽量模拟自然人的操作去页面搜索。账号和IP初始化是指定分配给各个Worker使用，状态存在数据库中，Worker初始化时候从数据库中读取账号IP信息。

1. **监控系统**

主要监控IP被封的情况，现象是账号在此IP上访问微博需要输入验证码或要绑定手机激活，当爬虫发现这些信息时候，将被封的IP和账号信息在数据库中打上标记。监控系统检测到被封的情况，通过短信等方式发给相应人员进行解封；这里需要开发一个使用简单的解封系统，因为如果IP被封很可能是普遍的事件，所使用的IP可能在一段时间内会统一被封掉。