设计微博系统中 “微博评论” 的高性能高可用计算架构。

1. 计算性能预估

【用户量】

2020.9月月活5.11亿，日活2.24亿 (参考《微博2020用户发展报告》)。

【关键行为】

评论微博

【行为建模 + 性能估算】

1. 由于之前算过，平均每天每人发1条微博 (只考虑文字微博)，则微博每天的发送量约为 2.5亿条。这里假设平均一条微博发评论的次数有50次，那么评论的请求量为 2.5亿 x 50 = 125 亿
2. 发评论的时间和看微博、发微博的时间基本重合，所以发评论的 TPS 和看微博差不多，为：125 亿 x 60% / (4 x 3600) ≈ 500K/s。（50万每秒）

2. 非热点事件时的高性能计算架构

【业务特性分析】

评论微博是写场景，但由于 TPS 很大，并且对于用户来说，发完评论后希望能够立刻被自己和大家看到，所以这里使用写缓冲来尽量避免评论丢失。

【架构分析】

1. 用户量过亿，应该要用多级负载均衡架构，覆盖 “DNS -> F5 -> Nginx -> 网关” 的4级负载均衡。
2. 发的评论放进Kafka 消息队列，然后发评论的服务器从消息队列里面读然后写入DB

【架构设计】

1. 负载均衡算法选择：发评论的时候依赖登录状态，登录状态一般都是保存在分布式缓存中的，因此发评论的时候，将请求发送给任意服务器都可以，这里选择“轮询”或者“随机”算法。
2. 业务服务器数量估算：发评论涉及几个关键的处理: 内容审核 (依赖审核系统)、数据写入存储 (依赖存储系统)、数据写入缓存 (依赖缓存系统)，因此按照一个服务每秒能处理500来估算，为了完成 500K/s 的 TPS，需要1000台服务器，加上一定的预留量，1100 台服务器就差不多了。

3. 由于发评论的架构和发微博的架构类似，只是多了个 kafka 消息队列当作写buffer；并且，发微博的 TPS 为10k/s （一万每秒），发评论的TPS 为 500k/s （50万每秒）， 所以可以将发评论业务service和发微博的业务service合并。

3. 热点事件时的高性能计算架构

造成热点事件的微博自己只有1～2条：

1. 这条微博可能会有几百万、甚至几千万的人围观
2. 这些用户在围观后还喜欢留下评论，假设有10%的围观用户会在事件发生后60分钟内评论。

由于很难预估到底哪个微博会是热点，也很难预估到底有多少人看热点微博，所以只能做好预防。

评论微博的思路与发微博相似，这里依然考虑对发出的评论进行限流，可以“漏桶算法”来存发出的评论，这里可以使用消息队列（限制100万）。