高并发高可用系统以及面试分析

1.高并发，高可用系统的一些思考

高并发依赖于场景和逻辑

不一定每个场景都会产生高并发，不要为了高并发而盲目的设计，**过度设计带来的问题远比意料之外的高并发要多很多**，依赖于具体场景和行为进行分析，一个购物类网站，抢购场景，会触发很多的读取商品详情，计算库存等操作，而且不需要每个请求都到达支付页面，也不会在网站主页带来很多的请求，所以需要针对抢购场景进行优化，而不是巨大的支付流程进行优化，当然商品数量多和用户多的情况，才需要也优化一下支付流程。

抛开场景，不谈流量的盲目高并发设计，一般是过度设计，未来维护臃肿而复杂的代码会惩罚你当初的过度设计。

突如其来的高并发

基本是被人刷了，或者比预估的情况要多了几倍，才有这种情况，前者可能性很大，最近这两年的金融网站，区块链，xx 币网站，基本会被羊毛党盯上，没做好访问防作弊，或者被对手 ddos 攻击，都会造成高并发中网站瘫痪，清洗流量一般就可以的，不要让辣鸡流量贯穿整个业务。**高并发会带来网站请求慢，但网站请求慢不一定是高并发了**，对症下药。

如果是网红带货什么的，给网站带来高并发了，**这种情况下可以在提前流量上做些手脚，例如：1. 用消息队列排队流量，例如猴王网站的抢购 2. 采用一些高端点的验证码，让一些秒杀抢购的机器号被过滤出去，尽量不影响真实用户的抢购结果 3. 分层过滤，例如静态资源放在 CDN 上，NGINX 层面使用自身的缓存，根据路由缓存基础数据，其他动态数据再读写数据库，而不是每个请求都经过整体的业务，这样带来的压力太大了，而且也不值得。**

并发中的整体项目稳定性

可以将高并发的业务部分分离到单独的服务器。这种情况下能够避免这部分业务带来机器性能的消耗，从而影响整个项目的稳定性，这样的结果不太好的。**也可以预先将热数据放到缓存中，提高读写效率，也能让 MySQL 比较稳定**，一般情况下能有高并发的网站，数据量也不少，举例某个电商网站，可能有 100w 商品数据，但大家抢购的是今天热推的 10 个商品，将这十个数据提取放到缓存中，而不是每次去数据库中查询，当然 MySQL 设置合理的话，自身的 buffer pool 也能搞定这些热数据缓存。这样相当于将这 10 条数据隔离出来，而不是

影响到整体数据。关于热点数据的发现，还有一些高端的是从 NGINX 访问日志层面实时分析，据说淘宝这种是这样的，实时发现访问很多的路由，分析路由获取到对应的商品数据，放到缓存中，减轻服务器压力。

稳定性不光存在于业务机器层面，也可能是网络宽带不够，或者程序在磁盘写日志的时候遇到瓶颈，或者内存不够，导致可用缓存空间很小。这些在测试和计算层面需要注意的问题，也会影响整体稳定性和性能的，应该提前解决。例如秒杀系统需要关注 CPU，并发读需要关注缓存，静态资源多的也需要考虑宽带。

预先测试

简单可以是压测关键业务部分，简单的查询一般不会带来问题，这个页面的静态资源太多，同时都在统一个域名下的话，相当于在现有并发的数量上加了更多，这非常不利，对于浏览器的加载也是不利的，简单业务也会体验很差。**压力测试应当遵循慢慢提升流量的方式，并发量和响应时间并不是等比例上涨的，慢慢提升并发量的测试，会展示代码的瓶颈部分在哪**，然后在去解决和提升。测试也可以

在基础的并发测试之外，还有在正是服务器上的全链路压力测试，为了防止和真实数据冲突，可以将请求中加上额外的标记，或者针对特定的用户帐号测试，在数据中体现这部分额外标记，测试完毕之后将数据删除。

同时，不要高并发的主要业务一直占用机器的 100% 的运算能力，这样整体逻辑和请求支持已经到了极限，基本再高一点就会造成问题，应该尽量让业务只占用机器 60-70% 的运算能力，留出一部分的余地，防止意外。

安全及备用方案

1. 从 NGINX 层面限制单个 IP 单位时间内请求频率和次数，屏蔽掉机器刷的可能性，从而不影响正常访问。 2. 切记高并发 + 高可用不可以用单点系统，例如不能因为热数据少而就用一台 Redis 服务器，或者更狠的直接本机缓存，一旦系统崩溃，相当于触发连锁反应，连保存现场和恢复都很难。 3. 提前设计兜底方案 ① **降级**，例如商品详情页面不展示推荐商品，或者减少推荐商品展示数量等， ② **限流**，不让更多流量涌入，能减少很多压力 ③ **过载临界点拒绝服务**，这个是最坏的情况，直接阻断压垮系统的最后一个流量。

2.面试官为什么会问你，如何设计一个高并发系统？

面试官心理分析

说实话，如果面试官问你这个题目，那么你必须要使出全身吃奶劲了。为啥？因为你没看到现在很多公司招聘的 JD 里都是说啥，有高并发就经验者优先。

如果你确实有真才实学，在互联网公司里干过高并发系统，那你确实拿 offer 基本如探囊取物，没啥问题。面试官也绝对不会这样来问你，否则他就是蠢。

假设你在某知名电商公司干过高并发系统，用户上亿，一天流量几十亿，高峰期并发量上万，甚至是十万。那么人家一定会仔细盘问你的系统架构，你们系统啥架构？怎么部署的？部署了多少台机器？缓存咋用的？MQ 咋用的？数据库咋用的？就是深挖你到底是如何扛住高并发的。

因为真正干过高并发的人一定知道，脱离了业务的系统架构都是在纸上谈兵，真正在复杂业务场景而且还高并发的时候，那系统架构一定不是那么简单的，用个 redis，用 mq 就能搞定？当然不是，真实的系统架构搭配上业务之后，会比这种简单的所谓“高并发架构”要复杂很多倍。

如果有面试官问你个问题说，如何设计一个高并发系统？那么不好意思，**一定是因为你实际上没干过高并发系统**。面试官看你简历就没啥出彩的，感觉就不咋地，所以就会问问你，如何设计一个高并发系统？其实说白了本质就是看看你有没有自己研究过，有没有一定的知识积累。

最好的当然是招聘个真正干过高并发的哥儿们咯，但是这种哥儿们人数稀缺，不好招。所以可能次一点的就是招一个自己研究过的哥儿们，总比招一个啥也不会的哥儿们好吧！

所以这个时候你必须得做一把个人秀了，秀出你所有关于高并发的知识！

面试题剖析

其实所谓的高并发，如果你要理解这个问题呢，其实就得从高并发的根源出发，为啥会有高并发？为啥高并发就很牛逼？

我说的浅显一点，很简单，就是因为刚开始系统都是连接数据库的，但是要知道数据库支撑到每秒并发两三千的时候，基本就快完了。所以才有说，很多公司，刚开始干的时候，技术比较 low，结果业务发展太快，有的时候系统扛不住压力就挂了。

当然会挂了，凭什么不挂？你数据库如果瞬间承载每秒 5000/8000，甚至上万的并发，一定会宕机，因为比如 mysql 就压根儿扛不住这么高的并发量。

所以为啥高并发牛逼？就是因为现在用互联网的人越来越多，很多 app、网站、系统承载的都是高并发请求，可能高峰期每秒并发量几千，很正常的。如果是什么双十一之类的，每秒并发几万几十万都有可能。

那么如此之高的并发量，加上原本就如此之复杂的业务，咋玩儿？真正厉害的，一定是在复杂业务系统里玩儿过高并发架构的人，但是你没有，那么我给你说一下你该怎么回答这个问题：

可以分为以下 6 点：

系统拆分
缓存

MQ

分库分表

读写分离

ElasticSearch

系统拆分

将一个系统拆分为多个子系统，用 dubbo 来搞。然后每个系统连一个数据库，这样本来就一个库，现在多个数据库，不也可以扛高并发么。

缓存

缓存，必须得用缓存。大部分的高并发场景，都是**读多写少**，那你完全可以在数据库和缓存里都写一份，然后读的时候大量走缓存不就得了。毕竟人家 redis 轻轻松松单机几万的并发。所以你可以考虑考虑你的项目里，那些承载主要请求的**读场景，怎么用缓存来抗高并发**。

MQ

MQ，必须得用 MQ。可能你还是会出现高并发写的场景，比如说一个业务操作里要频繁搞数据库几十次，增删改增删改，疯了。那高并发绝对搞挂你的系统，你要是用 redis 来承载写那肯定不行，人家是缓存，数据随时就被 LRU 了，数据格式还无比简单，没有事务支持。所以该用 mysql 还得用 mysql 啊。那你咋办？用 MQ 吧，大量的写请求灌入 MQ 里，排队慢慢玩儿，**后边系统消费后慢慢写**，控制在 mysql 承载范围之内。所以你得考虑考虑你的项目里，那些承载复杂写业务逻辑的场景里，如何用 MQ 来异步写，提升并发性。MQ 单机抗几万并发也是 ok 的，这个之前还特意说过。

分库分表

分库分表，可能到了最后数据库层面还是免不了抗高并发的要求，好吧，那么就将一个数据库拆分为多个库，多个库来扛更高的并发；然后将一个表**拆分为多个表**，每个表的数据量保持少一点，提高 sql 跑的性能。

读写分离

读写分离，这个就是说大部分时候数据库可能也是读多写少，没必要所有请求都集中在一个库上吧，可以搞个主从架构，**主库写**入，**从库读**取，搞一个读写分离。**读流量太多**的时候，还可以**加更多的从库**。

ElasticSearch

Elasticsearch，简称 es。es 是分布式的，可以随便扩容，分布式天然就可以支撑高并发，因为动不动就可以扩容加机器来扛更高的并发。那么一些比较简单的查询、统计类的操作，可以考虑用 es 来承载，还有一些全文搜索类的操作，也可以考虑用 es 来承载。

上面的 6 点，基本就是高并发系统肯定要干的一些事儿，大家可以仔细结合之前讲过的知识考虑一下，到时候你可以系统的把这块阐述一下，然后每个部分要注意哪些问题，之前都讲过了，你都可以阐述阐述，表明你对这块是有点积累的。

说句实话，毕竟你真正厉害的一点，不是在于弄明白一些技术，或者大概知道一个高并发系统应该长什么样？其实实际上在真正的复杂的业务系统里，做高并发要远远比上面提到的点要复杂几十倍到上百倍。你需要考虑：哪些需要分库分表，哪些不需要分库分表，单库单表跟分库分表如何 join，哪些数据要放到缓存里去，放哪些数据才可以扛住高并发的请求，你需要完成对一个复杂业务系统的分析之后，然后逐步逐步的加入高并发的系统架构的改造，这个过程是无比复杂的，一旦做过一次，并且做好了，你在这个市场上就会非常的吃香。

其实大部分公司，真正看重的，不是说你掌握高并发相关的一些基本的架构知识，架构中的一些技术，RocketMQ、Kafka、Redis、Elasticsearch，高并发这一块，你了解了，也只能是次一等的人才。对一个有几十万行代码的复杂的分布式系统，一步一步架构、设计以及实践过高并发架构的人，这个经验是难能可贵的。

2020 年 3 月 23 日 晚上 20-23 点 腾讯课堂 PHP 高并发 直播课

直播地址：https://ke.qq.com/course/328509?tuin=6

主讲老师介绍：

六星教育 Candy 老师：六星教育金牌老师，6 年项目开发发经验，曾就职于爱奇艺、亚信等大型互联网公司。精通各种主流框架。精通 linux 操作、mysql 的性能优化， 熟悉高并发解决方案。

晚上 20 点不见不散不许逃课呀 ，需要直播笔记、代码，视频以及往期录播，面试题，电子书籍进课堂找助理小姐姐领取啦