01 | 高并发系统：它的通用设计方法是什么？

来做个简单的比喻吧。·

从古至今，长江和黄河流域水患不断，远古时期大禹曾拓宽河道，清除淤沙让流水更加顺畅；都江堰作为史上最成功的治水案例之一，用引流将岷江之水分流到多个支流中，以分担水流压力；三门峡和葛洲坝通过建造水库将水引入水库先存储起来，然后再想办法把水库中的水缓缓地排出去，以此提高下游的抗洪能力。而我们在应对高并发大流量时也会采用类似“抵御洪水”的方案，归纳起来共有三种方法。

Scale-out（横向扩展）：分而治之是一种常见的高并发系统设计方法，采用分布式部署的方式把流量分流开，让每个服务器都承担一部分并发和流量。

缓存：使用缓存来提高系统的性能，就好比用“拓宽河道”的方式抵抗高并发大流量的冲击。

异步：在某些场景下，未处理完成之前我们可以让请求先返回，在数据准备好之后再通知请求方，这样可以在单位时间内处理更多的请求。

Scale-up vs Scale-out

Scale-up 通过购买性能更好的硬件来提升系统的并发处理能力，比方说目前系统 4 核 4G 每秒可以处理 200 次请求，那么如果要处理 400 次请求呢？很简单，我们把机器的硬件提升到 8 核 8G（硬件资源的提升可能不是线性的，这里仅为参考）

Scale-out 则是另外一个思路，它通过将多个低性能的机器组成一个分布式集群来共同抵御高并发流量的冲击。沿用刚才的例子，我们可以使用两台 4 核 4G 的机器来处理那 400 次请求。那么什么时候选择 Scale-up，什么时候选择 Scale-out 呢？

一般来讲，在我们系统设计初期会考虑使用 Scale-up 的方式，因为这种方案足够简单，所谓能用堆砌硬件解决的问题就用硬件来解决，但是当系统并发超过了单机的极限时，我们就要使用 Scale-out 的方式。

使用缓存提升性能

异步

异步处理异步也是一种常见的高并发设计方法，我们在很多文章和演讲中都能听到这个名词，与之共同出现的还有它的反义词：同步。比如分布式服务框架 Dubbo 中有同步方法调用和异步方法调用，IO 模型中有同步 IO 和异步 IO。

*那么什么是同步，什么是异步呢？*以方法调用为例，同步调用代表调用方要阻塞等待被调用方法中的逻辑执行完成。这种方式下，当被调用方法响应时间较长时，会造成调用方长久的阻塞，在高并发下会造成整体系统性能下降甚至发生雪崩。

异步调用恰恰相反，调用方不需要等待方法逻辑执行完成就可以返回执行其他的逻辑，在被调用方法执行完毕后再通过回调、事件通知等方式将结果反馈给调用方。

归根结底一句话：高并发系统的演进应该是循序渐进，以解决系统中存在的问题为目的和驱动力的。
我带着你了解了高并发系统设计的三种通用方法：Scale-out、缓存和异步。

这三种方法可以在做方案设计时灵活地运用，但它不是具体实施的方案，而是三种思想，在实际运用中会千变万化。就拿 Scale-out 来说，数据库一主多从、分库分表、存储分片都是它的实际应用方案。而我们需要注意的是，在应对高并发大流量的时候，系统是可以通过增加机器来承担流量冲击的，至于要采用什么样的方案还是要具体问题具体分析。