一、什么是高并发
二、如何提升系统的并发能力
2,1 常见的互联网分层架构
2.2 反向代理层的水平扩展
2.3 站点层的水平扩展
2.4 服务层的水平扩展
2.5 数据层的水平扩展
2.5.1 缓存
2.5.2 数据库

一、什么是高并发

高并发（High Concurrency）是互联网分布式系统架构设计中必须考虑的因素之一，它通常是指，通过设计保证系统能够同时并行处理很多请求。

高并发相关常用的一些指标有响应时间（Response Time），吞吐量（Throughput），每秒查询率QPS（Query Per Second），并发用户数等。

- 响应时间：系统对请求做出响应的时间。例如系统处理一个HTTP请求需要200ms，这个200ms就是系统的响应时间。
- 吞吐量：单位时间内处理的请求数量。
- QPS：每秒响应请求数。在互联网领域，这个指标和吞吐量区分的没有这么明显。
- 并发用户数：同时承载正常使用系统功能的用户数量。例如一个即时通讯系统，同时在线量一定程度上代表了系统的并发用户数。

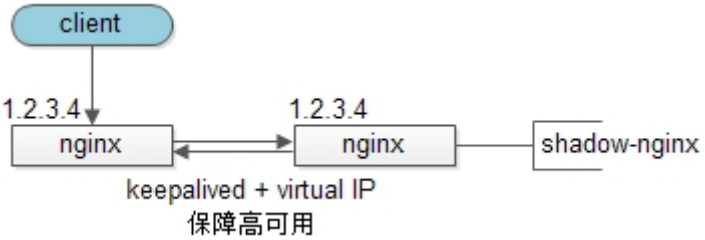
二、如何提升系统的并发能力

加机器的方式就不说了,花钱就完事了

2,1 常见的互联网分层架构

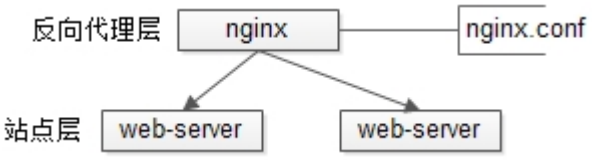
常见互联网分布式架构如上，分为：
（1）客户端层：典型调用方是浏览器browser或者手机应用APP
（2）反向代理层：系统入口，反向代理
（3）站点应用层：实现核心应用逻辑，返回html或者json
（4）服务层：如果实现了服务化，就有这一层
（5）数据-缓存层：缓存加速访问存储
（6）数据-数据库层：数据库固化数据存储

2.2 反向代理层的水平扩展



反向代理层的水平扩展，是通过“DNS轮询”实现的：dns-server对于一个域名配置了多个解析ip，**每次DNS解析请求来访问dns-server，会轮询返回这些ip。**
当nginx成为瓶颈的时候，只要增加服务器数量，新增nginx服务的部署，增加一个外网ip，就能扩展反向代理层的性能，做到理论上的无限高并发。
(有多个NG的情况下)

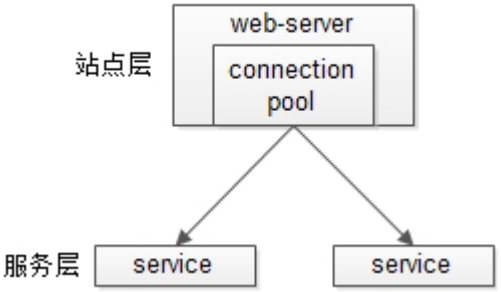
2.3 站点层的水平扩展



站点层的水平扩展，是通过“nginx”实现的。通过修改nginx.conf，**可以设置多个web后端。**

当web后端成为瓶颈的时候，只要增加服务器数量，新增web服务的部署，在nginx配置中配置上新的web后端，就能扩展站点层的性能，做到理论上的无限高并发。
(单个NG或者分发到具体某个NG时)

2.4 服务层的水平扩展



服务层的水平扩展，是通过“服务连接池”实现的。
站点层通过RPC-client调用下游的服务层RPC-server时，RPC-client中的连接池会建立与下游服务多个连接，当服务成为瓶颈的时候，只要增加服务器数量，新增服务部署，在RPC-client处建立新的下游服务连接，就能扩展服务层性能，做到理论上的无限高并发。**用上Eruka之类的服务注册发现功能**,使用多节点容灾

2.5 数据层的水平扩展

在数据量很大的情况下，数据层（**缓存，数据库**）涉及数据的水平扩展，将原本存储在一台服务器上的数据（缓存，数据库）水平拆分到不同服务器上去，以达到扩充系统性能的目的。

2.5.1 缓存

使用redis之类的缓存数据库,减少与数据库的交互

2.5.2 数据库

使用读写分离,分库分表等操作提高数据库的效率

> https://blog.csdn.net/weixin_42476601/article/details/82220027