# 代理

在编程语言中，**代理是指Delegate，也翻译为委托。就是将一件事委托给第三者，让第三者帮你完成。**

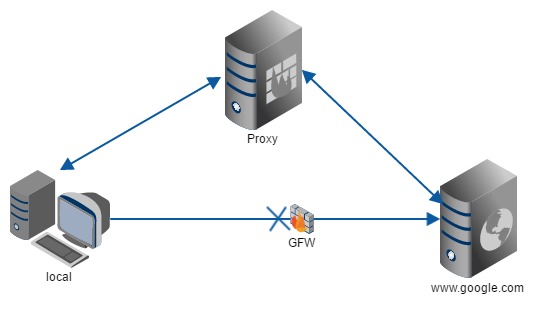
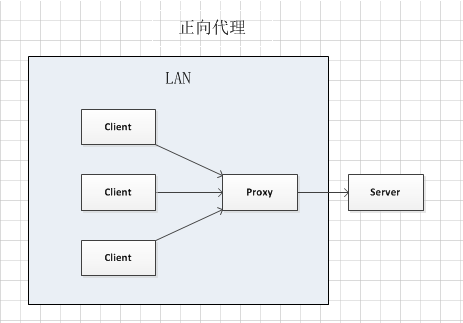
**「正向代理」代理的对象是客户端，也就是隐藏真实客户端，服务器不知道服务的对象是谁**

**「反向代理」代理的对象是服务端，也就是隐藏真实服务器端，客户端不知道是哪个服务器提供的服务**

## 正向代理

A同学在大众创业、万众创新的大时代背景下开启他的创业之路，目前他遇到的最大的一个问题就是启动资金，于是他决定去找马云爸爸借钱，可想而知，最后碰一鼻子灰回来了，情急之下，他想到一个办法，找关系开后门，经过一番消息打探，原来A同学的大学老师王老师是马云的同学，于是A同学找到王老师，托王老师帮忙去马云那借500万过来，当然最后事成了。不过马云并不知道这钱是A同学借的，马云是借给王老师的，最后由王老师转交给A同学。这里的王老师在这个过程中扮演了一个非常关键的角色，就是**代理**，也可以说是正向代理，王老师代替A同学办这件事，这个过程中，真正借钱的人是谁，马云是不知道的，这点非常关键。

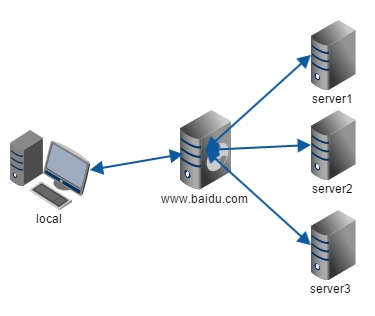
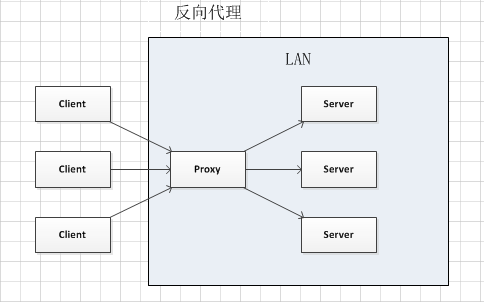
我们常说的代理也就是只正向代理，正向代理的过程，它隐藏了真实的请求客户端，服务端不知道真实的客户端是谁，客户端请求的服务都被代理服务器代替来请求，科学上网工具 Shadowsocks 扮演的就是典型的正向代理角色。在天朝用浏览器访问 www.google.com 时会被无情的墙掉，要想翻阅这堵墙，你可以在国外用 Shadowsocks 来搭建一台代理服务器，让代理帮我们去请求 www.google.com，代理再把请求响应结果再返回给我。

## 反向代理

大家都有过这样的经历，拨打10086 客服电话，一个地区的 10086 客服有几个或者几十个，你永远都不需要关心在电话那头的是哪一个，叫什么，男的，还是女的，漂亮的还是帅气的，你都不关心，你关心的是你的问题能不能得到专业的解答，你只需要拨通了10086 的总机号码，电话那头总会有人会回答你，只是有时慢有时快而已。那么这里的 10086 总机号码就是我们说的**反向代理**。客户不知道真正提供服务的人是谁。

反向代理隐藏了真实的服务端，当我们访问 www.baidu.com 的时候，就像拨打 10086 一样，背后可能有成千上万台服务器为我们服务，但具体是哪一台，你不知道，也不需要知道，你只需要知道反向代理服务器是谁就好了，www.baidu.com 就是我们的反向代理服务器，反向代理服务器会帮我们把请求转发到提供真实计算的服务器那里去。Nginx 就是性能非常好的反向代理服务器，它可以用来做负载均衡。

# 负载均衡

**负载均衡是由多台服务器以对称的方式组成一个服务器集合，每台服务器都具有等价的地位，都可以单独对外提供服务而无须其他服务器的辅助**。通过某种负载分担技术，将外部发送来的请求均匀分配到对称结构中的某一台服务器上，而接收到请求的服务器独立地回应客户的请求。均衡负载能够平均分配客户请求到服务器列阵，籍此提供快速获取重要数据，解决大量并发访问服务问题。

反向代理，是有把命令转发的能力，这个是必须基础

负载均衡，是把命令转发到不同的服务器上，均衡各个服务器

做了反向代理才能实现负载均衡。

* 分布式，**一个任务由多个人协作完成**。
  + 比如饭馆里有负责点菜的，有负责做菜的，有负责传菜的。
  + 比如饭馆里有买菜的，洗菜的，切菜的，炒菜的
  + 比如饭馆里有好几个负责做菜的
* 集群，**多个人紧密协作，来完成一个工作，就像一个人似的**
  + 比如饭馆里的所有厨师，或者所有工作人员可以看作一个集群
  + 集群有很多好处：
    - 是做菜的能力比一个人增加了
    - 多招一个厨子，就能轻易增加做菜的数量（**横向伸缩**），
    - 如果一个厨子病了，饭馆不至于停业（**高可用**）
* 负载均衡，**将一系列任务逐项分发给多个人，完成任务的能力增加，而且让大家都不闲着（提高资源利用率），减少任务等待的时间（降低延迟）**
  + 比如饭馆厨房里做菜的厨师有多个，新的菜来了让谁做呢？
    - 可以像发牌似的，一人一个轮流来（**轮流**）
    - 有人做的快，有人做的慢，可以按照一定的比例轮流（**加权轮流**）
    - 有可能有人累了干慢了，就会积压一堆订单，就根据谁面前的订单最少分给谁（**最少积压请求数**）

老王家5块田共50亩要耕，一个人干不完咋办？

* 于是叫来了大姨子、小姨子、小舅子、小叔子和9岁的小外甥5个人一起干。－－－ 这叫 ***集群***
* 可是，他家的地大小不一样，小姨子分的田最大，有20亩；小舅子个大小伙子，只有3亩，这怎么办？明显不“均衡”啊，得，不能按“块”分，要一人10亩地。－－－这叫 ***负载均衡***
* 又。。。又可是，人的能力大小不同，9岁的小外甥干的忒慢，小叔子干的很快，10亩地秒秒钟。这怎么办？ 按能力大小，分不同的地。能者多劳。－－－这叫 ***按权重负载均衡***

# 分布和集群

例1

分布式，**一个任务由多个人协作完成。**

* 比如饭馆里有负责点菜的，有负责做菜的，有负责传菜的。
* 比如饭馆里有买菜的，洗菜的，切菜的，炒菜的
* 比如饭馆里有好几个负责做菜的

比如一个任务（可以分成多个部分）由10个人负责完成，我们可以把这个任务分成10分，分别由这10个人去做，且这10个人的任务互不相关，各做各的 。当10个人都完成自己的事的时候，总的任务也就完成了。

集群，**多个人紧密协作，来完成一个工作，就像一个人似的**

* 比如饭馆里的所有厨师，或者所有工作人员可以看作一个集群

集群有很多好处：

* 是做菜的能力比一个人增加了
* 多招一个厨子，就能轻易增加做菜的数量（**横向伸缩**）
* 如果一个厨子病了，饭馆不至于停业（**高可用**）

比如工厂的生产流水线，分为三道程序，由三个员工负责完成。每个人负责自己面前的事。其中甲完成了工程的第一道工序后，生产线继续运行，到乙面前时，乙完成第二道工序，然后继续。三个人紧密合作，共同完成这样一道工序。

例2

如果一个任务由10个子任务组成，每个子任务单独执行需1小时，则在一台服务器上执行改任务需10小时。

采用分布式方案，提供10台服务器，每台服务器只负责处理一个子任务，不考虑子任务间的依赖关系，执行完这个任务只需一个小时。(这种工作模式的一个典型代表就是**[Hadoop](http://lib.csdn.net/base/hadoop" \t "_blank" \o "Hadoop知识库)**的Map/Reduce分布式计算模型）

而采用集群方案，同样提供10台服务器，每台服务器都能独立处理这个任务。假设有10个任务同时到达，10个服务器将同时工作，10小后，10个任务同时完成，这样，整身来看，还是1小时内完成一个任务！

## 分布式和集群相比较：

分布式就像电路中的并联电路，电流分开传输，一条路断了，则电流还可以通过另外的线路传输。即在分布式系统中，一个部分出现故障不影响其他部分工作，虽然最后任务没有完成。

集群就像电路中的串联电路，一旦任何一点线路断开，则电流就不能传输了。

简单来说，**分布式是以缩短单个任务的执行时间来提升效率的，而集群则是通过提高单位时间内执行的任务数来提升效率**。

# 集群

**1. 两大关键特性**

集群是一组协同工作的服务实体，用以提供比单一服务实体更具扩展性与可用性的服务平台。在客户端看来，一个集群就象是一个服务实体，但事实上集群由一组服务实体组成。与单一服务实体相比较，集群提供了以下两个关键特性：

· 可扩展性－－集群的性能不限于单一的服务实体，新的服务实体可以动态地加入到集群，从而增强集群的性能。

· 高可用性－－集群通过服务实体冗余使客户端免于轻易遇到out of service的警告。在集群中，同样的服务可以由多个服务实体提供。如果一个服务实体失败了，另一个服务实体会接管失败的服务实体。集群提供的从一个出 错的服务实体恢复到另一个服务实体的功能增强了应用的可用性。

**2. 两大能力**

为了具有可扩展性和高可用性特点，集群的必须具备以下两大能力：

· 负载均衡－－负载均衡能把任务比较均衡地分布到集群环境下的计算和网络资源。

· 错误恢复－－由于某种原因，执行某个任务的资源出现故障，另一服务实体中执行同一任务的资源接着完成任务。这种由于一个实体中的资源不能工作，另一个实体中的资源透明的继续完成任务的过程叫错误恢复。

负载均衡和错误恢复都要求各服务实体中有执行同一任务的资源存在，而且对于同一任务的各个资源来说，执行任务所需的信息视图（信息上下文）必须是一样的。

**3. 两大技术**

实现集群务必要有以下两大技术：

· 集群地址－－集群由多个服务实体组成，集群客户端通过访问集群的集群地址获取集群内部各服务实体的功能。具有单一集群地址（也叫单一影像）是集群的一个基 本特征。维护集群地址的设置被称为负载均衡器。负载均衡器内部负责管理各个服务实体的加入和退出，外部负责集群地址向内部服务实体地址的转换。有的负载均 衡器实现真正的负载均衡算法，有的只支持任务的转换。只实现任务转换的负载均衡器适用于支持ACTIVE-STANDBY的集群环境，在那里，集群中只有 一个服务实体工作，当正在工作的服务实体发生故障时，负载均衡器把后来的任务转向另外一个服务实体。

· 内部通信－－为了能协同工作、实现负载均衡和错误恢复，集群各实体间必须时常通信，比如负载均衡器对服务实体心跳测试信息、服务实体间任务执行上下文信息的通信。

具有同一个集群地址使得客户端能访问集群提供的计算服务，一个集群地址下隐藏了各个服务实体的内部地址，使得客户要求的计算服务能在各个服务实体之间分布。内部通信是集群能正常运转的基础，它使得集群具有均衡负载和错误恢复的能力。

## 集群分类

Linux集群主要分成三大类( 高可用集群， 负载均衡集群，科学计算集群)

* 高可用集群( High Availability Cluster)
* 负载均衡集群(Load Balance Cluster)
* 科学计算集群(High Performance Computing Cluster)

================================================

具体包括：

Linux High Availability 高可用集群 (普通两节点双机热备，多节点HA集群，RAC, shared, share-nothing集群等)

Linux Load Balance 负载均衡集群 (LVS等....)

Linux High Performance Computing 高性能科学计算集群 (Beowulf 类集群....)

分布式存储

其他类linux集群 (如Openmosix, rendering farm 等..)

## 详细介绍

**1. 高可用集群(High Availability Cluster)**

常见的就是2个节点做成的HA集群，有很多通俗的不科学的名称，比如"双机热备", "双机互备", "双机".

高可用集群解决的是保障用户的应用程序持续对外提供服务的能力。 (请注意高可用集群既不是用来保护业务数据的，保护的是用户的业务程序对外不间断提供服务，把因软件/硬件/人为造成的故障对业务的影响降低到最小程度)。

**2. 负载均衡集群(Load Balance Cluster)**

负载均衡系统：集群中所有的节点都处于活动状态，它们分摊系统的工作负载。一般Web服务器集群、数据库集群和应用服务器集群都属于这种类型。

负载均衡集群一般用于相应网络请求的网页服务器，数据库服务器。这种集群可以在接到请求时，检查接受请求较少，不繁忙的服务器，并把请求转到这些服务器上。从检查其他服务器状态这一点上看，负载均衡和容错集群很接近，不同之处是数量上更多。

**3. 科学计算集群(High Performance Computing Cluster)**

高性能计算(High Perfermance Computing)集群，简称HPC集群。这类集群致力于提供单个计算机所不能提供的强大的计算能力。

**高性能计算分类**

**高吞吐计算(High-throughput Computing)**

有一类高性能计算，可以把它分成若干可以并行的子任务，而且各个子任务彼此间没有什么关联。象在家搜寻外星人（ [SETI](mailto:SETI@HOME)[@HOME](http://my.oschina.net/xaxyf" \t "_blank) -- Search for Extraterrestrial Intelligence at Home ）就是这一类型应用。这一项目是利用Internet上的闲置的计算资源来搜寻外星人。SETI项目的服务器将一组数据和数据模式发给Internet上 参加SETI的计算节点，计算节点在给定的数据上用给定的模式进行搜索，然后将搜索的结果发给服务器。服务器负责将从各个计算节点返回的数据汇集成完整的 数据。因为这种类型应用的一个共同特征是在海量数据上搜索某些模式，所以把这类计算称为高吞吐计算。所谓的Internet计算都属于这一类。按照 Flynn的分类，高吞吐计算属于SIMD（Single Instruction/Multiple Data）的范畴。

**分布计算(Distributed Computing)**

另一类计算刚好和高吞吐计算相反，它们虽然可以给分成若干并行的子任务，但是子任务间联系很紧密，需要大量的数据交换。按照Flynn的分类，分布式的高性能计算属于MIMD（Multiple Instruction/Multiple Data）的范畴。

**4. 分布式（集群）与集群的联系与区别**

分布式是指将不同的业务分布在不同的地方。

而**集群指的是将几台服务器集中在一起，实现同一业务**。

**分布式中的每一个节点，都可以做集群。**

**而集群并不一定就是分布式的**。

举例：就比如新浪网，访问的人多了，他可以做一个群集，前面放一个响应服务器，后面几台服务器完成同一业务，如果有业务访问的时候，响应服务器看哪台服务器的负载不是很重，就将给哪一台去完成。

而**分布式，从窄意上理解，也跟集群差不多， 但是它的组织比较松散，不像集群，有一个组织性，一台服务器垮了，其它的服务器可以顶上来。**

**分布式的每一个节点，都完成不同的业务，一个节点垮了，哪这个业务就不可访问了。**

# 微服务

微服务与传统服务相比，最根本的区别在于服务的粒度。以前SOA提供的服务太大，可复用性不高。

微服务关注于具体的业务。在设计上考虑移植复用性和标准化。

微服务的优点：

1. 比如在微服务出现之前，我们创建了一个项目，包含了各个功能模块，比如订单、商品、售后、会员等。在开始的时候，这个项目可能很小很简洁。但是，成功的应用都会随着时间而变得巨大。为了适用于各种场景，或者增加点新的需求，都需要我们在项目里增加代码，也就是打一个补丁。在打补丁时，我们必须要了解之前代码的逻辑，对它有一个清晰的理解。如若不然，我们打的补丁可能就不够简洁。虽然也把窟窿填上了，但是可能原本一个小窟窿，我们却补了一大块上去。这样，一次两次可能还没什么，但是随着补丁的增多，代码肯定越来越丑陋，臃肿。而且，后面再增加代码，要理解前面的代码就变的更困难。毫无疑问，这是非常可怕的一件事。
2. 使用微服务，我们可以对项目进行拆分。比如一个电商项目，我们可以将各个模块拆分为单独的订单服务，商品服务，售后服务，会员服务……每个服务可以单独启动，然后通过组合又可以构成一个完整的系统。另外，当我们另一个项目业务中有订单部分时，我们可以直接将订单微服务拿过去使用。有更好的复用性。或者，到了双十一，然后店家搞促销，这段时间商城的订单肯定会大幅度增加。这时候我们就要考虑一个订单服务是不是不够啊，会不会压力太大导致系统崩溃啊。为了防止这种情况，我们可以再起一个订单微服务。这样即使订单量增加了，但我服务处理订单的能力也增加了。

# SpringBoot

SpringBoot是一个微框架。

SpringBoot的核心是Spring框架。它不是Spring的扩展或者补充，而是Spring的一个封装。Spring能做的，它也能能做。

SpringBoot的目的是为了简化Spring应用的初始搭建以及开发过程。

SpringBoot是一个非常好的微服务开发框架。

# SpringCloud

SpringCloud是一个微服务框架，提供的全套的分布式系统解决方案。

SpringCloud是一系列框架的有序集合。它将市场上比较成熟、经得起实际考验的服务框架组合起来，通过Spring Boot风格进行再封装，屏蔽掉了复杂的配置和实现原理，最终给开发者留出了一套简单易懂、易部署和易维护的分布式系统开发工具包

SpringCloud是在SpringBoot的基础上构建的。是用来快速构建微服务系统的工具集。

# Nginx

Nginx是一款轻量级、高性能的Web服务器、反向代理服务器及电子邮件（IMAP/POP3）代理服务器。

主要是代理服务器，适用于高并发的情况

Nginx由内核和模块组成，完成工作是通过查找配置文件将客户端请求映射到一个location block(location是用于URL匹配的命令)，location配置的命令会启动不同模块完成工作。

# FastDFS

分布式文件管理系统，文件上传时计算文件的哈希值，通过哈希值比较避免重复提交。