**负载均衡Load Balance学习笔记**

# 1. 简介

负载均衡是一种技术架构方法，它并不是具体指哪一种技术。

严格来说，负载平衡(Load balancing)是一种计算机网络技术，用来在多个计算机(计算机集群)、网络连接、CPU、磁盘驱动器或其他资源中分配负载，以达到最佳化资源使用、最大化吞吐率、最小化响应时间、同时避免过载的目的，使用带有负载平衡的多个服务器组件，取代单一的组件，可以通过冗余提高可靠性。负载平衡服务通常是由专用软体和硬件来完成。

# **2. 原理**

## 2.1 想解决的问题

单位时间内请求数，超过服务器处理请求能力。

场景： 如果一个用户的请求需要服务使用0.02秒来处理，那么该服务每秒只能处理50个这样的请求，每分钟也只能处理3000个。对于热门应用的常用功能，这个处理能力远远无法满足实际需求量。因此，我们就需要对该服务进行扩容。

## 2.2 解决途径

扩容主要分为Scale Up和Scale Out两种，分别对应“增强单个服务的服务能力”以及“增强服务数量”。

Scale Up较常规的操作就是为服务器加内存。但是服务器所能提供的能力实际上都受到其自身物理硬件能力以及价格昂贵的限制，因此我们就需要使用Scale Out方式将工作量分摊到多个服务器之中，如图1所示。

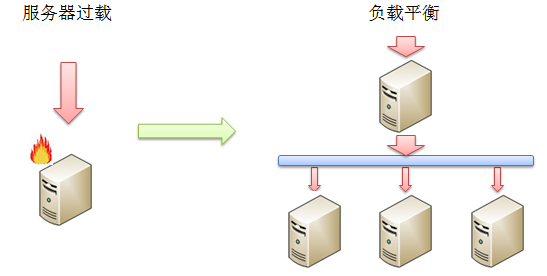


图1

如图1所示，当服务器的负载过多而来不及处理的时候，服务器就处于一种过载的状态。在该状态的服务常常会出现响应速度慢甚至无响应的情况。而在执行了Scale Out之后，我们将会使用多个服务器同时处理用户的请求。

## 2.3 Scale Out介绍

Scale Out需要使用一台特定的设备将这些请求分发到各个服务器。该设备会根据其内部所包含的请求分发逻辑来决定如何对这些请求进行分发，以避免出现单个服务器过载的情况。这些用来对请求进行分发的设备实际上就是**负载平衡服务器**。

一个拥有三台服务器的负载平衡系统，如果其中一台发生故障，那么负载平衡服务器可以通过向各个服务发送心跳等方式得知它们的异常，进而不再向这个发生了故障的服务器分发任务，如图2。

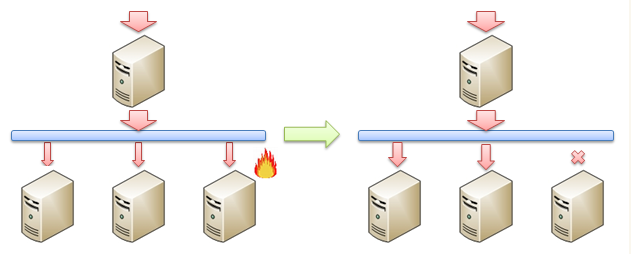


图2

若当前负载平衡系统所负担的服务容量超过了阈值，我们可以简单地通过添加服务器来解决这个问题，如图3。

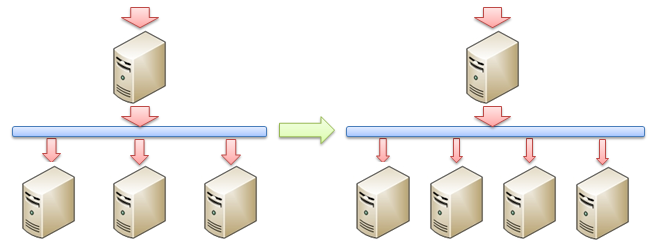


图3

# 应用场景

最常用的负载平衡解决方案主要分为三种：基于DNS的负载平衡，L3/4负载平衡（基于网络层的负载平衡），以及L7负载平衡（基于应用层的负载平衡）。在这些解决方案中，基于DNS的负载平衡是最简单、最早出现的方案。

## 3.1 基于DNS的负载平衡

### 3.1.1 应用场景

浏览器地址栏输入域名访问某网站时，浏览器先会查找本地DNS缓存是否拥有该域名所对应的IP地址。若有，浏览器使用该IP地址访问该网站的内容；若无，浏览器将向DNS发送一个请求，以获得该域名所对应的IP并添加到本地DNS缓存中。

而在DNS中，一个域名可能和多个IP地址绑定。在这种情况下，DNS响应将会按照Round Robin方式返回这些IP地址的列表。例如在多次通过nslookup或host等命令来查看特定域名所对应的IP时，其可能的返回如下（因国内网络原因，您需要FQ再进行试验）：

$ host -t a google.com

　　google.com has address 72.14.207.99

　　google.com has address 64.233.167.99

　　google.com has address 64.233.187.99

　　$ host -t a google.com

　　google.com has address 64.233.187.99

　　google.com has address 72.14.207.99

　　google.com has address 64.233.167.99

很明显，不同的DNS请求所返回的结果会按照Round Robin进行轮换，进而使得不同的用户访问不同的IP地址，平衡各个服务器的负载。

**小贴士：Dns**

Dns是域名系统 (Domain Name System) 的缩写，它是由解析器和域名服务器组成的。域名服务器是指保存有该网络中所有主机的域名和对应IP地址，并具有将域名转换为IP地址功能的服务器。其中域名必须对应一个IP地址，而IP地址不一定有域名。

域名系统采用类似目录树的等级结构。域名服务器为客户机/服务器模式中的服务器方，它主要有两种形式：主服务器和转发服务器。将域名映射为IP地址的过程就称为“域名解析”。在Internet上域名与IP地址之间是一对一（或者多对一）的，域名虽然便于人们记忆，但机器之间只能互相认识IP地址，它们之间的转换工作称为域名解析，域名解析需要由专门的域名解析服务器来完成，DNS就是进行域名解析的服务器。

DNS 命名用于 Internet 等 TCP/IP 网络中，通过用户友好的名称查找计算机和服务。当用户在应用程序中输入 DNS 名称时，DNS 服务可以将此名称解析为与之相关的其他信息，如 IP 地址。因此，你在上网时输入的网址，是通过域名解析系统解析找到了相对应的IP地址，这样才能上网。其实，域名的最终指向是IP

**小贴士：Round Robin**

### 3.1.1 弊端

致命的缺点：为了减少DNS请求的次数以提高访问效率，浏览器常常缓存了DNS查询的结果。如果一个IP的服务失效了，浏览器仍会根据DNS缓存中所记录的信息向该不可用的服务发送请求（不同的浏览器可能有不同的行为）。虽然说整个服务只有一处IP所对应的服务失效了，但是从用户的角度看来该网站已经不可访问。

因此基于DNS的负载平衡方案并不能作为一个独立的负载平衡解决方案来提供高可用性的保障，而是作为其它负载平衡解决方案的补充方案来使用。

## 3.2 L3/4负载平衡（网络层）

### 3.2.1 简介

L3/4指的就是负载平衡服务器会根据OSI模型中的第三层网络层（Network Layer）和第四层传输层（Transport Layer）所包含的数据来进行负载平衡操作。在这种负载平衡服务器中，这些数据主要包含数据包的IP头和TCP、UDP等协议的协议头，如图4。

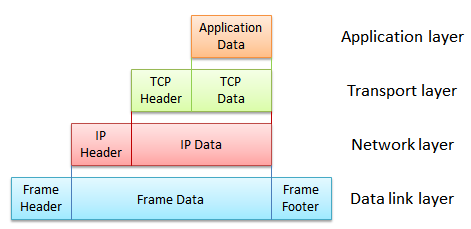


图4

### 3.2.2 工作原理

在数据到达时，负载平衡服务器将根据自身算法以及OSI模型三四层所包含的数据决定需要处理该数据的服务并将其转发。

负载平衡的运行包含三方面内容：

• 知道当前有效的服务到底有哪些

• 根据自身的分派算法决定分派给哪个服务

• 将数据发送到目标服务上

**确定服务的有效性。**为了保证从负载平衡服务器所派发的数据包能被服务器集群正常处理，负载平衡服务器需要周期性地发送状态查询请求以探测服务正在有效地工作。这种状态查询请求常常会超出很多人的认知：如果服务崩溃但是承载它的操作系统正常工作，那么该操作系统仍然会正常响应负载平衡服务器所发出的Ping命令，只是此时TCP连接会失败；如果服务并没有崩溃，而只是挂起，那么它仍然可以接受TCP连接，只是无法接收HTTP请求。

以上状态查询请求实际上是特定于服务的具体实现，因此很多负载平衡服务器都允许用户添加自定义脚本以执行特定于服务的查询。这些状态查询请求常常包含了很多测试，甚至会尝试从服务中返回数据。

一旦负载平衡服务器发现其所管理的某个服务不再有效，那么它就不会再将任何数据转发给该服务，直到该服务回归正常状态。在这种情况下，其它各个服务就需要分担失效服务器原本承担的工作。

**注意：**在某个服务失效后，整个系统应该仍具有足够的总容量以处理负载。

**负载均衡算法。**最常见的误区：负载平衡服务器会根据各个服务的响应速度或负载状况来决定请求需要到达的服务。实际是根据负载平衡算法计算结果来分配的。常用算法：Round Robin算法、哈希算法等。

**负载平衡服务器转发方式。**主要分为三种：Direct routing，Tunnelling以及IP address translation。

## 3.3 L7负载平衡（应用层）

主要通过OSI模型中的第七层应用层中的数据决定如何分发负载，如图4。

相较于L3/4负载平衡服务所使用的数据，L7负载平衡服务所使用的应用层数据更贴近服务本身，因此其具有更精确的负载平衡行为。

### 3.3.1 工作原理

运行时，L7负载平衡服务器上的操作系统会将接收到的各个数据包组织成为用户请求，并根据在该请求中所包含的的数据来决定分配给哪个服务，如图5。

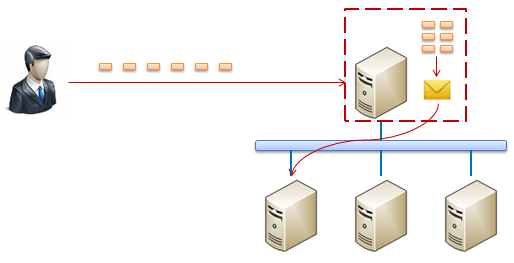


图5

**Cookie。** L3/4负载平衡对于某些具有关联关系的请求，会根据某些算法（如计算IP的哈希值）来决定处理该请求的服务。这种方法并不是很稳定。当一个服务失效或用户的IP发生变化的时候，用户与服务之间的对应关系就将发生改变。此时用户原有的会话数据在新的服务上并不存在，进而导致一系列问题。

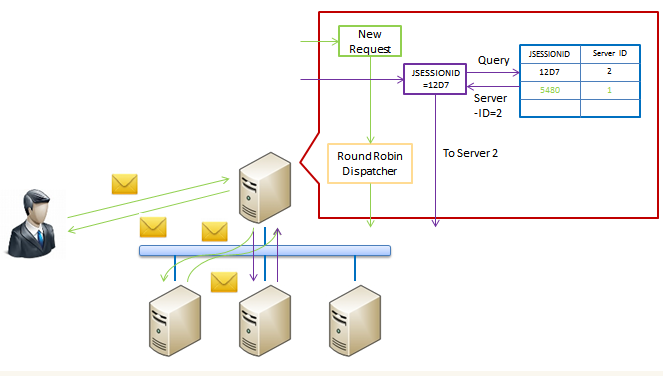
其实产生以上的最根本原因就是用户与服务之间的关联关系是通过某些外部环境创建的，而非由用户/服务本身来管理。因此它不能抵御外部环境变化的冲击。如果要在用户和服务之间建立稳定的关联关系，那么就需要一种稳定的在用户和服务实例之间传递的数据。在Web服务中，这种数据就是Cookie。

**基于Cookie的负载平衡服务，实际上就是分析用户请求中的某个特定Cookie，并根据其值决定需要分发到的目标地址。**其主要分为两种方式：Cookie Learning以及Cookie Insertion。

### 3.3.2 Cookie Learning

Cookie Learning是不具有侵入性的一种解决方案。通过分析用户与服务实例通讯过程中所传递的Cookie来决定如何分派负载。

在用户与服务第一次通讯时，负载平衡服务找不到相应的Cookie，会根据负载平衡算法分配到某个服务上，而在服务返回时，负载平衡服务器将会把对应的Cookie以及服务的地址记录在负载平衡服务器中。当用户再次与服务通讯时，负载平衡服务器就会根据Cookie中所记录的数据找到前一次服务该用户的服务实例，并将请求转发到该服务上。如图6。

 图6

**弊端。**

**1、**对高可用性的支持很差。如果一旦负载平衡服务器失效，那么在该负载平衡服务器上所维护的Cookie和服务之间的匹配关系将全部丢失。这样当备份负载平衡服务器启动之后，所有的用户请求都将被定向到随机的服务。

2、会话维护功能对内存的消耗。与L3/4服务器上的会话维护不同，一个Cookie的失效时间可能非常长，至少在一次用户使用中可能会持续几个小时。对于一个访问量达到每秒上万次的系统而言，负载平衡服务器需要维护非常多的会话，甚至可能会将服务器的内存消耗殆尽。反过来，如果将负载平衡服务器中的Cookie过期时间设置得太短，那么当用户重新访问负载平衡服务器的时候，其将被导向到一个错误的服务。

### 3.3.2 Cookie Insertion

通过向响应中添加Cookie以记录被分派到的服务实例，并在下一次处理请求时根据该Cookie所保存的值来决定分发到的服务实例。

在用户与服务器进行第一次通讯的时候，负载平衡服务器会根据负载平衡算法为该请求分配一个服务。在接收到服务所返回的数据之后，负载平衡服务器将会向响应中插入一个Cookie，以记录该服务的ID。当用户再次发送请求到负载平衡服务器时，其将根据该Cookie里所记录的服务ID派发该请求。如图7。

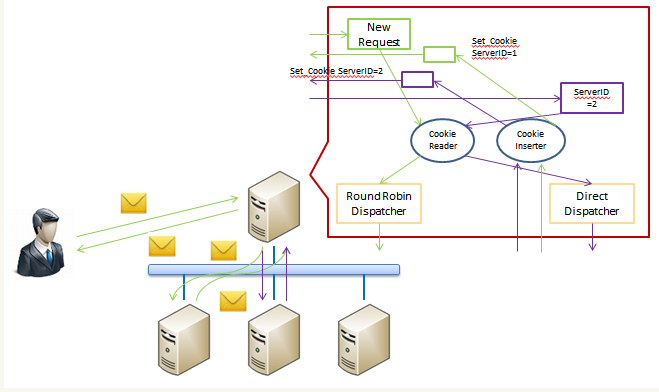


图7