

负载均衡架构



静修佛缘 (/u/d6e5ad19fcef) + 关注

2017.08.09 17:38* 字数 9128 阅读 1004 评论 0 喜欢 21

(/u/d6e5ad19fcef)

【摘要】

面对大量用户访问、高并发请求，海量数据，可以使用高性能的服务器、大型数据库，存储设备，高性能Web服务器，采用高效率的编程语言比如(Go,Scala)等，当单机容量达到极限时，我们需要考虑业务拆分和分布式部署，来解决大型网站访问量，并发量高，海量数据的问题。

从单机网站到分布式网站，很重要的区别是业务拆分和分布式部署，将应用拆分后，部署到不同的机器上，实现大规模分布式系统。分布式和业务拆分解决了，从集中到分布的问题，但是每个部署的独立业务还存在单点的问题和访问统一入口问题，为解决单点故障，我们可以采取冗余的方式。将相同的应用部署到多台机器上。解决访问统一入口问题，我们可以在集群前面增加负载均衡设备，实现流量分发。

负载均衡（Load Balance），意思是将负载（工作任务，访问请求）进行平衡、分摊到多个操作单元（服务器，组件）上进行执行。是解决高性能，单点故障（高可用），扩展性（水平伸缩）的终极解决方案。

1. 负载均衡原理

系统的扩展可分为纵向（垂直）扩展和横向（水平）扩展。纵向扩展，是从单机的角度通过增加硬件处理能力，比如CPU处理能力，内存容量，磁盘等方面，实现服务器处理能力的提升，不能满足大型分布式系统（网站），大流量，高并发，海量数据的问题。因此需要采用横向扩展的方式，通过添加机器来满足大型网站服务的处理能力。比如：一台机器不能满足，则增加两台或者多台机器，共同承担访问压力。这就是典型的集群和负载均衡架构，如下图：

```
graph LR
    User[用户] -- 输入URL --> Browser[浏览器]
    Browser --> LB[负载均衡设备]
    LB -- 请求分发 --> AppCluster[应用集群]
    subgraph AppCluster
        direction TB
        A[淘宝网A]
        B[淘宝网B]
    end
```

常见的负载均衡方案：

https://www.jianshu.com/p/8f7242cbf469?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=reco...

1/20

应用集群：将同一应用部署到多台机器上，组成处理集群，接收负载均衡设备分发的请求，进行处理，并返回相应数据。

负载均衡设备：将用户访问的请求，根据负载均衡算法，分发到集群中的一台处理服务器。（一种把网络请求分散到一个服务器集群中的可用服务器上去的设备）

```
(/apps/  
utm_sc  
banner
```

负载均衡的作用（解决的问题）：

- (1) 解决并发压力，提高应用处理性能（增加吞吐量，加强网络处理能力）；
- (2) 提供故障转移，实现高可用；
- (3) 通过添加或减少服务器数量，提供网站伸缩性（扩展性）；
- (4) 安全防护（负载均衡设备上做一些过滤，黑白名单等处理）；

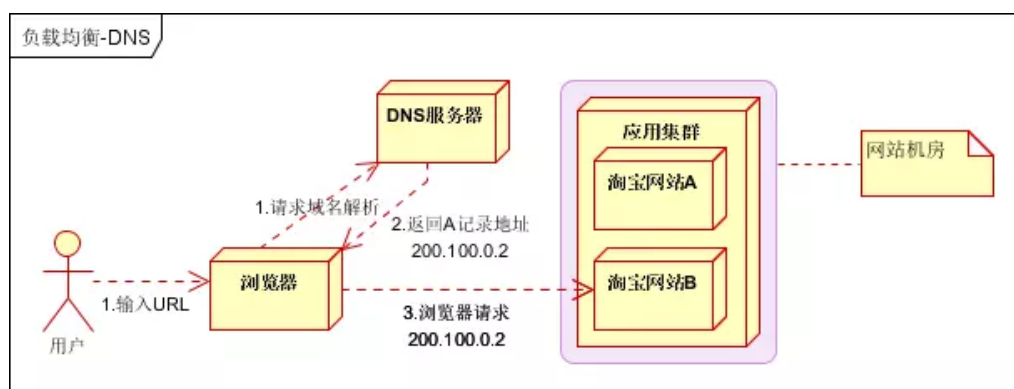
2. 负载均衡分类

根据实现技术不同，可分为**DNS负载均衡**，**HTTP负载均衡**，**IP负载均衡**，**链路层负载均衡**等。

(<https://click.ycombinator.com/slot=309fa8-45d0b0863984>)

2.1 DNS负载均衡

最早的负载均衡技术，利用域名解析实现负载均衡，在DNS服务器，配置多个A记录，这些A记录对应的服务器构成集群。大型网站总是部分使用DNS解析，作为第一级负载均衡。如下图：



优点：

- （1）**使用简单：**负载均衡工作，交给DNS服务器处理，省掉了负载均衡服务器维护的麻烦；
- （2）**提高性能：**可以支持基于地址的域名解析，解析成距离用户最近的服务器地址，可以加快访问速度，改善性能；

缺点：

- (1) **可用性差：** DNS解析是多级解析，新增/修改DNS后，解析时间较长；解析过程中，用户访问网站将失败；
- (2) **扩展性低：** DNS负载均衡的控制权在域名商那里，无法对其做更多的改善和扩展；
- (3) **维护性差：** 也不能反映服务器的当前运行状态；支持的算法少；不能区分服务器的差异（不能根据系统与服务的状态来判断负载）

实践建议：

将DNS作为第一级负载均衡，A记录对应着内部负载均衡的IP地址，通过内部负载均衡将请求分发到真实的Web服务器上。一般用于互联网公司，复杂的业务系统不合适使用。
如下图：

(/apps/
utm_sc
banner

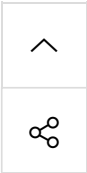
(https:/
click.y
slot=30
9fa8-4
5d0b0!
863984

2.2 IP负载均衡

在网络层通过修改请求目标地址进行负载均衡。

用户请求数据包，到达负载均衡服务器后，负载均衡服务器在操作系统内核进程获取网络数据包，根据负载均衡算法得到一台真实服务器地址，然后将请求目的地址修改为，获得的真实ip地址，不需要经过用户进程处理。

真实服务器处理完成后，响应数据包回到负载均衡服务器，负载均衡服务器，再将数据包源地址修改为自身的ip地址，发送给用户浏览器。如下图：



优点：

- (1) 在内核进程完成数据分发，比在应用层分发性能更好；

缺点：

- (1) 所有请求响应都需要经过负载均衡服务器，集群最大吞吐量受限于负载均衡服务器网卡带宽；

2.3 链路层负载均衡

在通信协议的数据链路层修改mac地址，进行负载均衡。

数据分发时，不修改ip地址，只修改目标mac地址，配置真实物理服务器集群所有机器虚拟ip和负载均衡服务器ip地址一致，达到不修改数据包的源地址和目标地址，进行数据分发的目的。

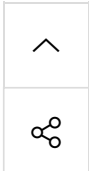
实际处理服务器ip和数据请求目的ip一致，不需要经过负载均衡服务器进行地址转换，可将响应数据包直接返回给用户浏览器，避免负载均衡服务器网卡带宽成为瓶颈。也称为直接路由模式（DR模式）。如下图：

优点：性能好；

缺点：配置复杂；

实践建议：DR模式是目前使用最广泛的一种负载均衡方式。

2.4 混合型负载均衡



由于多个服务器群内硬件设备、各自的规模、提供的服务等差异，可以考虑给每个服务器群采用最合适的负载均衡方式，然后又在这多个服务器群间再一次负载均衡或群集起来以一个整体向外界提供服务（即把这多个服务器群当做一个新的服务器群），从而达到最佳的性能。将这种方式称之为混合型负载均衡。

此种方式有时也用于单台均衡设备的性能不能满足大量连接请求的情况下。是目前大型互联网公司，普遍使用的方式。

方式一，如下图：

以上模式适合有动静分离的场景，反向代理服务器（集群）可以起到缓存和动态请求分发的作用，当时静态资源缓存在代理服务器时，则直接返回到浏览器。如果动态页面则请求后面的应用负载均衡（应用集群）。

方式二，如下图：

以上模式，适合动态请求场景。因混合模式，可以根据具体场景，灵活搭配各种方式，以上两种方式仅供参考。

3. 负载均衡算法

常用的负载均衡算法有：轮询，随机，最少链接，源地址散列，加权等方式；

3.1 轮询

将所有请求，依次分发到每台服务器上，适合服务器硬件同相同的场景。

(https://
click.ye
slot=30
9fa8-4i
5d0b0!
863984



- 优点：服务器请求数目相同；

缺点：服务器压力不一样，不适合服务器配置不同的情况；

(/apps/
utm_sc
banner

3.2 随机

请求随机分配到各个服务器。

- 优点：使用简单；

缺点：不适合机器配置不同的场景；

3.3 最少链接

将请求分配到连接数最少的服务器（目前处理请求最少的服务器）。

- 优点：根据服务器当前的请求处理情况，动态分配；

缺点：算法实现相对复杂，需要监控服务器请求连接数；

(https:/
click.y
slot=30
9fa8-4i
5d0b0!
863984

3.4 Hash（源地址散列）

<t>根据IP地址进行Hash计算，得到IP地址。

- 优点：将来自同一IP地址的请求，同一会话期内，转发到相同的服务器；实现会话粘滞。

缺点：目标服务器宕机后，会话会丢失；

3.5 加权

在轮询，随机，最少链接，Hash等算法的基础上，通过加权的方式，进行负载服务器分配。

- 优点：根据权重，调节转发服务器的请求数目；

缺点：使用相对复杂；



4. 硬件负载均衡

采用硬件的方式实现负载均衡，一般是单独的负载均衡服务器，价格昂贵，一般土豪级公司可以考虑，业界领先的有两款，F5和A10。

使用硬件负载均衡，主要考虑一下几个方面：

(1) **功能考虑：**功能全面支持各层级的负载均衡，支持全面的负载均衡算法，支持全局负载均衡；

(2) **性能考虑：**一般软件负载均衡支持到5万级并发已经很困难了，硬件负载均衡可以支持

(3) **稳定性：**商用硬件负载均衡，经过了良好的严格的测试，从经过大规模使用，在稳定性方面高；

(4) **安全防护：**硬件均衡设备除具备负载均衡功能外，还具备防火墙，防DDOS攻击等安全功能；

(5) **维护角度：**提供良好的维护管理界面，售后服务和技术支持；

(6) **土豪公司：**F5 Big Ip 价格：15w~55w不等；A10 价格：55w-100w不等；

缺点：

(1) 价格昂贵；

(2) 扩展能力差；

一般硬件的负载均衡也要做双机高可用，因此成本会比较高。互联网公司一般使用开源软件，因此大部分应用采用软件负载均衡；部分采用硬件负载均衡。比如某互联网公司，目前是使用几台F5做全局负载均衡，内部使用Nginx等软件负载均衡。

5 Ngnix负载均衡

Ngnix是一款轻量级的Web服务器/反向代理服务器，工作在七层Http协议的负载均衡系统。具有高性能、高并发、低内存使用等特点。是一个轻量级的Http和反向代理服务器。Ngnix使用epoll and kqueue作为开发模型。能够支持高达 50,000 个并发连接数的响应。

操作系统：Liunix, Windows (Linux、FreeBSD、Solaris、Mac OS X、AIX以及Microsoft Windows)

开发语言：C

(/apps/
utm_sc
banner

(https:/
click.y
slot=30
9fa8-4i
5d0b0!
863984



并发性能：官方支持每秒5万并发，实际国内一般到每秒2万并发，有优化到每秒10万并发的。具体性能看应用场景。

(/apps/
utm_sc
banner

5.1 特点

- (1) **模块化设计**：良好的扩展性，可以通过模块方式进行功能扩展。
- (2) **高可靠性**：主控进程和worker是同步实现的，一个worker出现问题，会立刻启动另一个worker。
- (3) **内存消耗低**：一万个长连接（keep-alive）,仅消耗2.5MB内存。
- (4) **支持热部署**：不用停止服务器，实现更新配置文件，更换日志文件、更新服务器程序版本。
- (5) **并发能力强**：官方数据每秒支持5万并发；
- (6) **功能丰富**：优秀的反向代理功能和灵活的负载均衡策略；

5.2 功能

基本功能：

- (1) 支持静态资源的web服务器。
- (2) http,smtp,pop3协议的反向代理服务器、缓存、负载均衡；
- (3) 支持FASTCGI（fpm）
- (4) 支持模块化，过滤器（让文本可以实现压缩，节约带宽）,ssl及图像大小调整。
- (5) 内置的健康检查功能
- (6) 基于名称和ip的虚拟主机
- (7) 定制访问日志
- (8) 支持平滑升级
- (9) 支持KEEPALIVE
- (10) 支持url rewrite
- (11) 支持路径别名
- (12) 支持基于IP和用户名的访问控制。

(https:/
click.ye
slot=30
9fa8-4i
5d0b0!
863984



(13) 支持传输速率限制，支持并发数限制。

Nginx的高并发，官方测试支持5万并发连接。实际生产环境能到2-3万并发连接数。10000个非活跃的HTTP keep-alive 连接仅占用约2.5MB内存。三万并发连接下，10个Nginx进程，消耗内存150M。淘宝tengine团队测试结果是“24G内存机器上，处理并发请求可达200万”。

(/apps/
utm_sc
banner

5.3 架构

5.3.1 基本工作模式

一个master进程，生成一个或者多个worker进程。但是这里master是使用root身份启动的，因为nginx要工作在80端口。而只有管理员才有权限启动小于1023的端口。master主要是负责的作用只是启动worker，加载配置文件，负责系统的平滑升级。其它的工作是交给worker。那么当worker被启动之后，也只是负责一些web最简单的工作，而其他的工作都是有worker中调用的模块来实现的。

模块之间是以流水线的方式实现功能的。流水线，指的是一个用户请求，由多个模块组合各自的功能依次实现完成的。比如：第一个模块只负责分析请求首部，第二个模块只负责查找数据，第三个模块只负责压缩数据，依次完成各自工作。来实现整个工作的完成。

他们是如何实现热部署的呢？其实是这样的，我们前面说master不负责具体的工作，而是调用worker工作，他只是负责读取配置文件，因此当一个模块修改或者配置文件发生变化，是由master进行读取，因此此时不会影响到worker工作。在master进行读取配置文件之后，不会立即的把修改的配置文件告知worker。而是让被修改的worker继续使用老的配置文件工作，当worker工作完毕之后，直接干掉这个子进程，更换新的子进程，使用新的规则。

5.3.2 sendfile机制

(https://
click.ye
slot=30
9fa8-4i
5d0b0!
863984



Sendfile机制，用户将请求发给内核，内核根据用户的请求调用相应用户进程，进程在处理时需要资源。此时再把请求发给内核（进程没有直接IO的能力），由内核加载数据。内核查找到数据之后，会把数据复制给用户进程，由用户进程对数据进行封装，之后交给内核，内核在进行tcp/ip首部的封装，最后再发给客户端。这个功能用户进程只是发生了一个封装报文的过程，却要绕一大圈。因此nginx引入了sendfile机制，使得内核在接收到数据之后，不再依靠用户进程给予封装，而是自己查找自己封装，减少了一个很长一段时间浪费，这是一个提升性能的核心点。

以上内容摘自网友发布文章，简单一句话是资源的处理，直接通过内核层进行数据传递，避免了数据传递到应用层，应用层再传递到内核层的开销。

目前高并发的处理，一般都采用sendfile模式。通过直接操作内核层数据，减少应用与内核层数据传递。

5.3.3 通信模型（I/O复用机制）

开发模型：epoll和kqueue。

支持的事件机制：kqueue、epoll、rt signals、/dev/poll、event ports、select以及poll。

支持的kqueue特性包括EV_CLEAR、EV_DISABLE、NOTE_LOWAT、EV_EOF，可用数据的数量，错误代码。

支持sendfile、sendfile64和sendfilev;文件AIO；DIRECTIO;支持Accept-filters和TCP_DEFER_ACCEP.

以上概念较多，大家自行百度或谷歌，知识领域是网络通信（BIO,NIO,AIO）和多线程方面的知识。

5.4 均衡策略

nginx的负载均衡策略可以划分为两大类：内置策略和扩展策略。内置策略包含加权轮询和ip hash，在默认情况下这两种策略会编译进nginx内核，只需在nginx配置中指明参数即可。扩展策略有很多，如fair、通用hash、consistent hash等，默认不编译进nginx内核。由于在nginx版本升级中负载均衡的代码没有本质性的变化，因此下面将以nginx1.0.15稳定版为例，从源码角度分析各个策略。

5.4.1 加权轮询 (weighted round robin)

轮询的原理很简单，首先我们介绍一下轮询的基本流程。如下是处理一次请求的流程图：

(/apps/
utm_sc
banner

图中有两点需要注意：

第一，如果可以把加权轮询算法分为先深搜索和先广搜索，那么nginx采用的是先深搜索算法，即将首先将请求都分给高权重的机器，直到该机器的权值降到了比其他机器低，才开始将请求分给下一个高权重的机器；

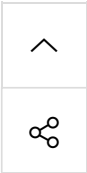
(https://
click.y
slot=30
9fa8-4
5d0b0!
863984

第二，当所有后端机器都down掉时，nginx会立即将所有机器的标志位清成初始状态，以避免造成所有的机器都处在timeout的状态，从而导致整个前端被夯住。

5.4.2 ip hash

ip hash是nginx内置的另一个负载均衡的策略，流程和轮询很类似，只是其中的算法和具体的策略有些变化，如下图所示：

5.4.3 fair



fair策略是扩展策略，默认不被编译进nginx内核。其原理是根据后端服务器的响应时间判断负载情况，从中选出负载最轻的机器进行分流。这种策略具有很强的自适应性，但是实际的网络环境往往不是那么简单，因此要慎用。

5.4.4 通用hash、一致性hash

这两种也是扩展策略，在具体的实现上有些差别，通用hash比较简单，可以以nginx内置的变量为key进行hash，一致性hash采用了nginx内置的一致性hash环，可以支持memcache。

5.5 场景

Ngnix一般作为入口负载均衡或内部负载均衡，结合反向代理服务器使用。以下架构示例，仅供参考，具体使用根据场景而定。

5.5.1 入口负载均衡架构

Ngnix服务器在用户访问的最前端。根据用户请求再转发到具体的应用服务器或二级负载均衡服务器（LVS）。

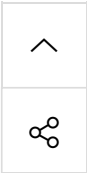
5.5.2 内部负载均衡架构

LVS作为入口负载均衡，将请求转发到二级Ngnix服务器，Ngnix再根据请求转发到具体的应用服务器。

5.5.3 Ngnix高可用

(/apps/
utm_sc
banner

(https:/
click.y
slot=30
9fa8-4i
5d0b0!
863984



分布式系统中，应用只部署一台服务器会存在单点故障，负载均衡同样有类似的问题。一般可采用主备或负载均衡设备集群的方式节约单点故障或高并发请求分流。

Ngnix高可用，至少包含两个Ngnix服务器，一台主服务器，一台备服务器，之间使用Keepalived做健康监控和故障检测。开放VIP端口，通过防火墙进行外部映射。

DNS解析公网的IP实际为VIP。

6 LVS负载均衡

LVS是一个开源的软件，由毕业于国防科技大学的章文嵩博士于1998年5月创立，用来实现Linux平台下的简单负载均衡。LVS是Linux Virtual Server的缩写，意思是Linux虚拟服务器。

基于IP层的负载均衡调度技术，它在操作系统核心层上，将来自IP层的TCP/UDP请求均衡地转移到不同的服务器，从而将一组服务器构成一个高性能、高可用的虚拟服务器。

操作系统：Liunx

开发语言：C

并发性能：默认4096，可以修改但需要重新编译。

6.1 功能

LVS的主要功能是实现IP层（网络层）负载均衡，有NAT,TUN,DR三种请求转发模式。

6.1.1 LVS/NAT方式的负载均衡集群

NAT是指Network Address Translation，它的转发流程是：Director机器收到外界请求，改写数据包的目标地址，按相应的调度算法将其发送到相应Real Server上，Real Server处理完该请求后，将结果数据包返回到其默认网关，即Director机器上，Director机器再改写数据包的源地址，最后将其返回给外界。这样就完成一次负载调度。

构架一个最简单的LVS/NAT方式的负载均衡集群Real Server可以是任何的操作系统，而且无需做任何特殊的设定，惟一要做的就是将其默认网关指向Director机器。Real Server可以使用局域网的内部IP(192.168.0.0/24)。Director要有两块网卡，一块网卡绑定

(https://
click.ye
slot=30
9fa8-4;
5d0b0!
863984



一个外部IP地址 (10.0.0.1)，另一块网卡绑定局域网的内部IP(192.168.0.254)，作为Real Server的默认网关。

LVS/NAT方式实现起来最为简单，而且Real Server使用的是内部IP，可以节省Real IP的开销。但因为执行NAT需要重写流经Director的数据包，在速度上有一定延迟；

当用户的请求非常短，而服务器的回应非常大的情况下，会对Director形成很大压力，成为新的瓶颈，从而使整个系统的性能受到限制。

6.1.2 LVS/TUN方式的负载均衡集群

TUN是指IP Tunneling，它的转发流程是：Director机器收到外界请求，按相应的调度算法,通过IP隧道发送到相应Real Server，Real Server处理完该请求后，将结果数据包直接返回给客户。至此完成一次负载调度。

最简单的LVS/TUN方式的负载均衡集群架构使用IP Tunneling技术，在Director机器和Real Server机器之间架设一个IP Tunnel，通过IP Tunnel将负载分配到Real Server机器上。Director和Real Server之间的关系比较松散，可以在是同一个网络中，也可以是在不同的网络中，只要两者能够通过IP Tunnel相连就行。收到负载分配的Real Server机器处理完后会直接将反馈数据送回给客户，而不必通过Director机器。实际应用中，服务器必须拥有正式的IP地址用于与客户机直接通信，并且所有服务器必须支持IP隧道协议。

该方式中Director将客户请求分配到不同的Real Server，Real Server处理请求后直接回应给用户，这样Director就只处理客户机与服务器的一半连接，极大地提高了Director的调度处理能力，使集群系统能容纳更多的节点数。另外TUN方式中的Real Server可以在任何LAN或WAN上运行，这样可以构筑跨地域的集群，其应对灾难的能力也更强，但是服务器需要为IP封装付出一定的资源开销，而且后端的Real Server必须是支持IP Tunneling的操作系统。

6.1.3 LVS/DR方式的负载均衡集群

DR是指Direct Routing，它的转发流程是：Director机器收到外界请求，按相应的调度算法将其直接发送到相应Real Server，Real Server处理完该请求后，将结果数据包直接返回给客户，完成一次负载调度。

构架一个最简单的LVS/DR方式的负载均衡集群Real Server和Director都在同一个物理网段中，Director的网卡IP是192.168.0.253，再绑定另一个IP： 192.168.0.254作为对外界的virtual IP，外界客户通过该IP来访问整个集群系统。Real Server在lo上绑定IP：192.168.0.254，同时加入相应的路由。

LVS/DR方式与前面的LVS/TUN方式有些类似，前台的Director机器也是只需要接收和调度外界的请求，而不需要负责返回这些请求的反馈结果，所以能够负载更多的Real Server，提高Director的调度处理能力，使集群系统容纳更多的Real Server。但LVS/DR需要改写请求报文的MAC地址，所以所有服务器必须在同一物理网段内。

6.2 架构

LVS架设的服务器集群系统有三个部分组成：最前端的负载均衡层（Load Balancer），中间的服务器群组层，用Server Array表示，最底层的数据共享存储层，用Shared Storage表示。在用户看来所有的应用都是透明的，用户只是在使用一个虚拟服务器提供的高性能服务。

LVS的体系架构如图：

LVS的各个层次的详细介绍：

Load Balancer层：位于整个集群系统的最前端，有一台或者多台负载调度器（Director Server）组成，LVS模块就安装在Director Server上，而Director的主要作用类似于一个路由器，它含有完成LVS功能所设定的路由表，通过这些路由表把用户的请求分发给Server Array层的应用服务器（Real Server）上。同时，在Director Server上还要安装对Real Server服务的监控模块Ldirectord，此模块用于监测各个Real Server服务的健康状况。在Real Server不可用时把它从LVS路由表中剔除，恢复时重新加入。

Server Array层：由一组实际运行应用服务的机器组成，Real Server可以是WEB服务器、MAIL服务器、FTP服务器、DNS服务器、视频服务器中的一个或者多个，每个Real Server之间通过高速的LAN或分布在各地的WAN相连接。在实际的应用中，Director Server也可以同时兼任Real Server的角色。

Shared Storage层：是为所有Real Server提供共享存储空间和内容一致性的存储区域，在物理上，一般有磁盘阵列设备组成，为了提供内容的一致性，一般可以通过NFS网络文件系统共享数据，但是NFS在繁忙的业务系统中，性能并不是很好，此时可以采用集群文件系统，例如Red hat的GFS文件系统，oracle提供的OCFS2文件系统等。

从整个LVS结构可以看出，Director Server是整个LVS的核心，目前，用于Director Server的操作系统只能是Linux和FreeBSD，linux2.6内核不用任何设置就可以支持LVS功能，而FreeBSD作为Director Server的应用还不是很多，性能也不是很好。对于Real Server，几乎可以是所有的系统平台，Linux、windows、Solaris、AIX、BSD系列都能很好的支持。



6.3 均衡策略

LVS默认支持八种负载均衡策略，简述如下：

(1) 轮询调度 (Round Robin)

调度器通过“轮询”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接数和系统负载。

(2) 加权轮询 (Weighted Round Robin)

调度器通过“加权轮询”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器能处理更多的访问流量。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

(3) 最少链接 (Least Connections)

调度器通过“最少连接”调度算法动态地将网络请求调度到已建立的连接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能，采用“最小连接”调度算法可以较好地均衡负载。

(4) 加权最少链接 (Weighted Least Connections)

在集群系统中的服务器性能差异较大的情况下，调度器采用“加权最少链接”调度算法优化负载均衡性能，具有较高权值的服务器将承受较大比例的活动连接负载。调度器可以自动问询真实服务器的负载情况，并动态地调整其权值。

(5) 基于局部性的最少链接 (Locality-Based Least Connections)

“基于局部性的最少链接”调度算法是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器；若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用“最少链接”的原则选出一个可用的服务器，将请求发送到该服务器。

(6) 带复制的基于局部性最少链接 (Locality-Based Least Connections with Replication)

“带复制的基于局部性最少链接”调度算法也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个目标IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服务器组，按“最小连接”原则从服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载，则按“最小连接”原则从这个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

(/apps/
utm_sc
banner

(https:/
click.y
slot=3(
9fa8-4;
5d0b0!
863984



(7) 目标地址散列 (Destination Hashing)

“目标地址散列”调度算法根据请求的目标IP地址，作为散列键 (Hash Key) 从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

(8) 源地址散列 (Source Hashing)

“源地址散列”调度算法根据请求的源IP地址，作为散列键 (Hash Key) 从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

除具备以上负载均衡算法外，还可以自定义均衡策略。

6.4 场景

一般作为入口负载均衡或内部负载均衡，结合反向代理服务器使用。相关架构可参考Ngnix场景架构。

7 HaProxy负载均衡

HAProxy也是使用较多的一款负载均衡软件。HAProxy提供高可用性、负载均衡以及基于TCP和HTTP应用的代理，支持虚拟主机，是免费、快速并且可靠的一种解决方案。特别适用于那些负载特大的web站点。运行模式使得它可以很简单安全的整合到当前的架构中，同时可以保护你的web服务器不被暴露到网络上。

7.1 特点

支持两种代理模式：TCP（四层）和HTTP（七层），支持虚拟主机；

配置简单，支持url检测后端服务器状态；

做负载均衡软件使用，在高并发情况下，处理速度高于nginx；

TCP层多用于Mysql从（读）服务器负载均衡。（对Mysql进行负载均衡，对后端的DB节点进行检测和负载均衡）

能够补充Nginx的一些缺点比如Session的保持，Cookie引导等工作；

7.2 均衡策略

支持四种常用算法：

- (1) roundrobin：轮询，轮流分配到后端服务器；
- (2) static-rr：根据后端服务器性能分配；



- (3) **leastconn**：最小连接者优先处理；
- (4) **source**：根据请求源IP，与Nginx的IP_Hash类似。

(/apps/
utm_sc
banner

小礼物走一走，来简书关注我

赞赏支持

📖 架构设计 (/nb/15197826) 举报文章 © 著作权归作者所有




静修佛缘 (/u/d6e5ad19fcef)
写了 48092 字，被 60 人关注，获得了 98 个喜欢
(/u/d6e5ad19fcef)

+ 关注

喜欢 | 21



更多分享




下载简书 App ▶
随时随地发现和创作内容




(https://
click.y
slot=30
9fa8-4;
5d0b0!
863984

(/apps/redirect?utm_source=note-bottom-click)


被以下专题收入，发现更多相似内容



架构设计 (/c/e84d216fb043?utm_source=desktop&utm_medium=notes-included-collection)




Java技术升华 (/c/d83d00973774?utm_source=desktop&utm_medium=notes-included-collection)

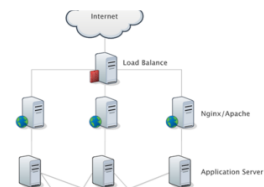


收藏 (/c/2461dce333b9?utm_source=desktop&utm_medium=notes-included-collection)

^



(/p/fc268327d299?



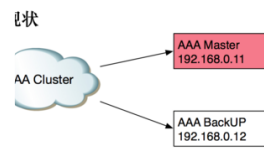
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend
分布式架构实践——负载均衡 (/p/fc268327d299?utm_campaign=maleski...

分布式架构实践——负载均衡 也许当我老了，也一样写代码；不为别的，只为了爱好。 1 什么是负载均衡
(Load balancing) 在网站创立初期，我们一般都使用单台机器对台提供集中式服务，但是随着业务量越...

Bobby0322 (/u/99ee95856388?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

(/p/735303aae25e?



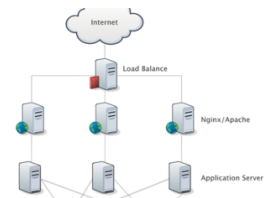
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend
基于LVS的AAA负载均衡架构实践 (/p/735303aae25e?utm_campaign=ma...

概要 本次分享将从一次实际的负载均衡改造案例出发，通过介绍项目背景、选型思路、测试方法和问题分析
等方面展开，总结负载均衡架构的一般套路和经验教训。 一、背景 项目背景是某企业的AAA管理系统，...

RiboseYim (/u/8cc1dba4bc96?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

(/p/b576d146d653?



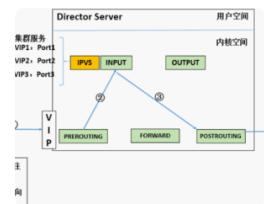
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend
LVS负载均衡 (/p/b576d146d653?utm_campaign=maleskine&utm_conte...

本文部分观点图片采用于：http://chenx1242.blog.51cto.com 随着智能机的逐渐普及，大量的APP应用使的
现在生活越来越方便。基本上在完成一部手机走天下。而在大规模互联网应用中源于互联网应用的高并发...

BossHuang (/u/b2c09db564de?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

(/p/374118659663?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend
LVS-负载均衡原理 (/p/374118659663?utm_campaign=maleskine&utm_c...

负载均衡集群是 load balance 集群的简写，翻译成中文就是负载均衡集群。常用的负载均衡开源软件有
nginx、lvs、haproxy，商业的硬件负载均衡设备F5、Netscale。这里主要是学习 LVS 并对其进行了详细的...


jiangmo (/u/de31051e96e1?

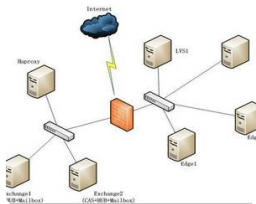
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

(/p/511aaf37802e?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

浅谈大型网站之负载均衡架构 (/p/511aaf37802e?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation) (/apps/utms/banner

概念 负载均衡，英文名称为Load Balance，其意思就是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，共同分担客户负载，降低服务器压力，避免单一服务器 overload，从而有效提高业务系统的可靠性和可用性。

 Java技术栈 (/u/25c50c845c4f?)



utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

虚弱 (/p/c23a92389c72?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

 一路李花开 (/u/a91198ab49ab?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

实体店要采用不同的思路 (/p/7c7b4e0100ed?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)


机会往往就隐藏在危机当中，眼下现在服装生意这么难做，说明到了一个转折时期，就看谁能想出解决方法，也许过了两年后，发现，原来要是这么干，才是出路呀。 谁也想不到具体的、稳定的操作方法，都是...

 妙裁 (/u/12e2c99e3001?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

笨办法学Python ex17 (/p/1bce6b3bcec7?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

更多文件操作 输入： 遇到的问题 笨办法里用的是苹果系统直接创建了一个测试文件，win系统的话，这边采取了上一个教程教的新建一个测试文件并写入内容，然后再复制内容到新的文件当中，结果基本和教程一...

 Joemini (/u/ff2cf99b99d7?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation


(/p/66d21be91caa?)



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

家庭艾灸，到底怎么做才能更简单更有效？ (/p/66d21be91caa?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

养生，为的是有更好的身体，有更好的身体，为的是更好的享受生活！ 艾江山，致力于推广家庭艾灸养生，希望家庭艾灸能更简单，更容易坚持，更有效。 那么，究竟怎么做，才能达到我们的目标呢？今天来试试...

 宁and静 (/u/ae647727ac39?)

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation

