

基于 Nginx 的负载均衡技术与优化

戴伟¹, 马明栋², 王得玉²

(1. 南京邮电大学 通信与信息工程学院, 江苏 南京 210003;

2. 南京邮电大学 地理与生物信息学院, 江苏 南京 210003)

摘要: 随着互联网的普及和智能上网设备的高速发展, 网络用户和业务量呈指数增长, 热点事件的到来更是会引发网络节点的波动。传统的单一的网络服务器根本无法承担大量并发业务请求, 因此服务器集群技术应运而生。为了能在服务器集群中合理分配业务, 使各个服务器都发挥应有的性能, 负载均衡机制及均衡算法成为了关键。Nginx 作为一款轻量级高并发的 Web 服务器, 单机可以承受十万级的并发请求, 而其模块化的设计, 更是可以方便地配置为反向代理服务器, 将请求分配给上游服务器。文中将分析 Nginx 的反向代理优势, 通过对其自带的负载均衡算法进行分析, 并优化出一种具有实时反馈能力的负载均衡算法。通过测试, 改进后的算法分配更加合理, 处理连接的速度也更加快速, 满足了设计要求。

关键词: Nginx; 负载均衡; 反向代理; IO 复用; 动态反馈; 服务器集群

中图分类号: TP39

文献标识码: A

文章编号: 1673-629X(2019)03-0077-04

doi: 10.3969/j.issn.1673-629X.2019.03.016

Research and Optimization of Load Balancing Based on Nginx

DAI Wei¹, MA Ming-dong², WANG De-yu²

(1. School of Telecommunications & Information Engineering, Nanjing University of Posts and

Telecommunications, Nanjing 210003, China;

2. School of Geographical and Biological Information, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: With the popularization of the Internet and the rapid development of smart Internet devices, the number of network users and services has grown exponentially, and the arrival of hot-spot events will cause network nodes to fluctuate. The traditional single network server cannot handle a large number of concurrent service requests at all, therefore the server cluster technology arises. In order to distribute the service reasonably in the server cluster and make each server play its performance, load balancing mechanism and balancing algorithm become the key. As a lightweight and highly concurrent Web server, Nginx can withstand 100 000 concurrent requests on a single machine, and its modular design allows it to be easily configured as a reverse proxy server to allocate requests to upstream servers. In this paper, we analyze the advantages of Nginx reverse proxy and its own load balancing algorithm, and optimize a load balancing algorithm with real-time feedback. Through testing, the improved algorithm is more reasonable in allocation and faster in connection processing, which meets the design requirements.

Key words: Nginx; load balancing; reverse proxy; IO multiplexing; dynamic feedback; server cluster

0 引言

随着计算机和智能手机的迅速发展, 中国互联网群体上网的方式愈发多样, 体量也逐渐巨大, 这自然提高了 Web 服务器的设计要求。传统的单台服务器, 很难承受大量用户的并发请求, 如若宕机则整个服务器就会崩溃, 只有将多台服务器协同起来作为一个集群,

相互配合, 才能持续稳定地为用户服务, 也会避免出现因某一个服务器崩溃, 而导致整个服务无法继续的情形。那么对一个集群的多个服务器之间的工作进行协调, 使它们都在其接受范围内发挥其最大性能是必须面对的问题。负载均衡技术便是用来将外部的请求, 均匀地分配到集群的各个服务器中, 进行单独处理再

收稿日期: 2018-04-08

修回日期: 2018-08-15

网络出版时间: 2018-12-19

基金项目: 江苏省自然科学基金-青年基金项目(BK20140868)

作者简介: 戴伟(1994-), 男, 硕士研究生, 研究方向为网络编程与服务器端开发; 马明栋, 博士, 教授, 研究方向为地理信息系统平台软件设计与开发等; 王得玉, 博士, 副教授, 研究方向为水环境遥感、GIS 软件设计与开发。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20181219.1511.038.html>

回发,以此快速处理数据,在支持高并发请求的同时保持高效的服务。

Nginx 作为一款高性能的 Web 服务器,采用多进程的基于事件驱动的架构,全异步的网络 I/O 处理机制,以及极少的进程间切换设计,使得它能够同时支持百万级别的 TCP 连接^[1]。同时其自带的 upstream 模块除了可以没有任何阻塞地实现 Nginx 与上游服务器的交互外,还实现了转发上游应用层协议的响应包体到下游客户端的功能。这些设计都确保 Nginx 可以作为一个优秀的反向代理服务器,通过负载均衡将服务均衡协调地转发给 Apache、Tomcat 等上游服务器,顺利完成高并发的 HTTP 请求和响应。

文中在当前负载均衡技术的基础上,根据 Nginx 作为反向代理服务器的优越性能,分析了 Nginx 自身所提供的负载均衡算法的不足,通过对负载均衡算法进行优化改进,实现高可靠、高性能的服务器设计,以提高用户的网络体验。

1 Nginx 分析

1.1 Nginx 架构

作为一款轻量级的 Web 服务器,Nginx 可以在处理高并发请求的同时保持高效的性能,这也是其应用广泛的关键^[2]。传统的 Apache 服务器每个进程一次只能处理一个请求,当面对大量的并发连接请求时,Apache 只能增加更多的进程数,这将导致服务器产生成百上千的进程,而进程之间的切换会带来大量的系统资源消耗,从而降低了服务器处理速度。Nginx 采用的是一个 master 进程管理多个 worker 进程的方法。master 进程不处理用户请求,而用来管理 worker 进程。当某个 worker 进程出现异常关闭后,会负责建立新的进程,以保证 Nginx 工作的稳定性。worker 进程处理请求,多个 worker 进程间相互独立,避免了在处理并发请求时同步锁的设置,减少了资源消耗,提高了服务器性能。

通常将 work 进程数设置为服务器 CPU 核心数,这样减少了进程间无谓的切换,充分发挥了 SMP 多核 CPU 架构,真正实现了并发处理。在工作进程中,Nginx 采用 IO 多路复用的事件驱动型设计,全异步的网络 IO 处理机制,能够单机处理十万级的 TCP 并发连接。Nginx 的工作模式如图 1 所示。

1.2 Nginx 反向代理

Nginx 对外表现为服务器,当接受到客户端的 HTTP 请求时,并不产生响应,但也不同于 Squid 等其他服务器直接转发,而是会将用户发来的请求缓存一份,然后再通过负载均衡算法将完整的请求缓存从它的上游集群服务器中选择合适的进行转发,而上游服

务器处理请求生成的响应会直接转发给 Nginx,此时 Nginx 会边接受边将响应转发给客户,而不是完整地缓存到响应再一并转发。工作流程如图 2 所示。

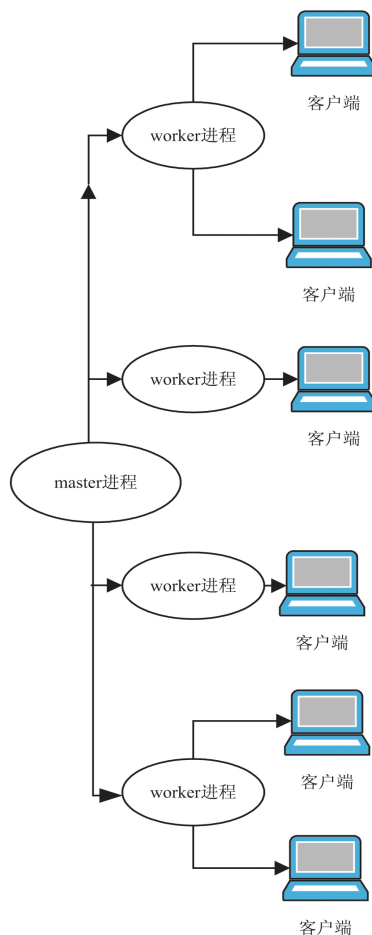


图 1 Nginx 工作模式

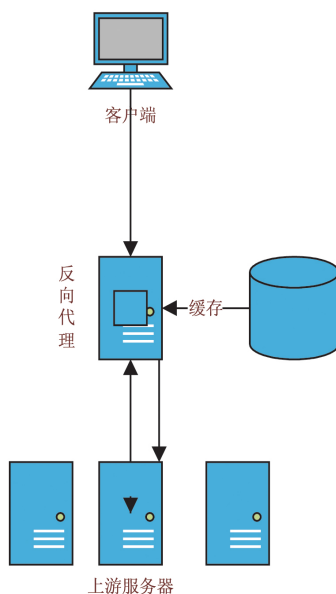


图 2 Nginx 反向代理示意

这正是 Nginx 作为反向代理服务器的优势所在,通常客户端与代理服务之间的网络为公网,其网络环境更加复杂,请求处理的时间会持续很久才能完成。

通过缓存完整的 HTTP 请求包,再进行转发以减少上游服务器的处理时间及负担;而代理服务器与上游服务器之间有专网专线相通,请求传输更加快速。这样的边处理边转发方式减少了上游服务器并发压力,更好地处理了高并发请求下的服务器压力问题。同时,这种设计还保护了上游服务器的安全。客户端与代理服务器之间通过公网相连,保证正常的公网客户的访问需求。因为上游服务器存在于专有网络中,连接必须通过反向代理才可向上进行访问。对于那些试图对服务器进行恶意攻击的连接,代理服务器会进行屏蔽,不将其转发到上游,保证了连接的安全。

2 Nginx 负载均衡

2.1 Nginx 常规负载均衡算法

Nginx 主要通过其所提供的 upstream 模块,配置负载均衡的算法,实现客户端 IP 到后端服务器的负载均衡^[3]。

目前 Nginx 模块所支持的调度算法主要有 5 类:

(1) 轮询及加权轮询: 每个请求按时间顺序逐一分配到不同的后端服务器,如果某台服务器宕机,则故障系统自动剔除,用户访问不受影响。加权值 Weight 越大,则分配的访问几率越高。主要是在后端的每个服务器的性能都相当的情况下。

(2) ip_hash: 每个请求按访问 IP 的 hash 结果分配,这样来自同一个 IP 的访客可以固定到同一个后端服务器,有效地解决了动态网页存在的 session 共享问题。

(3) url_hash: 此方法按访问的 url 的 hash 结果来分配请求,使固定的 url 定向到同一个后端服务器,可以进一步提高后端缓存服务器的效率^[4]。Nginx 自身不支持 url_hash,需要安装 Nginx 的 hash 软件包。

(4) least_conn: 最少连接数法,均衡器记录目前所有活跃连接,把下一个新的请求发给当前含有最少连接数的节点。该算法针对 TCP 连接进行,但由于不同应用对系统资源的消耗可能差异很大,而连接数无法反映出真实的应用负载,因此在使用重型 Web 服务器作为集群节点服务时,该算法在均衡负载的效果上要大打折扣。为了减少这种不利的影响,可以对每个节点设置最大的连接数上限,通常通过阈值设定体现。

(5) fair: 该方法可以根据页面大小和加载时间长短智能地进行负载均衡,也就是根据后端服务器的响应时间来分配请求,响应时间短的优先分配。

上述算法中,轮询和加权轮询为静态算法,优点在于设计简单,运行快,同时不会给服务器带来额外负担,但是该算法是理想化的,没有考虑服务器在实际运行时遇到的特殊情况,实际可靠性较差^[5]。最小连

接算法和 fair 算法为动态算法,与静态算法相比有性能提升。该类动态算法虽然考虑服务器的响应快慢、连接数大小等实时运行情况,但要么考虑因素过于单一,要么对于集群中的服务器性能以同样化对待,没有处理异构情况,最后的性能提升自然也不够明显^[6]。

所以,在进行负载均衡算法设计的过程中,需要考虑几个基本点:

(1) 在保证平衡的基础上,算法尽量设计的简单。若是为了能够均衡分配,考虑的因素过多,只会适得其反,给服务器带来额外的计算压力^[7]。

(2) 考虑到集群中的各个服务器的性能不同,可以承受的并发量也不同。必须对异构的服务集群设定不同的分配权重。

(3) 在上述的基础上,将其实时负载情况考虑在内,如:服务器当前资源使用率、响应时间、实时连接数等信息作为计算权重式的因子^[8]。根据服务器当前的负载情况,进行请求分配。实现动态的兼有预测的实时负反馈算法。

2.2 优化算法

针对 Nginx 自带负载均衡算法的不足,基于最少连接算法,提出带有动态反馈的新型改进算法。其思想主要是在分派任务前,先根据各个服务器的自身性能,设置权值优先级^[9];之后在每次负载均衡时,通过对当前服务器的负载率和连接数进行分析,动态地更新分配权值,每次择优分配,权值越高越优先分配,实现性能优化,负载均衡^[10]。

考虑在一个集群中,服务器的集合为 $S_i = \{S_1, S_2, \dots, S_n\} (n > 1)$,服务器的固有性能为 $C_i = \{C_1, C_2, \dots, C_n\} (n > 1)$ 。则可以给每台服务器设置的初始权值为:

$$W_i = C_i / \sum_{i=1}^n C_i \quad (1 < i < n) \quad (1)$$

针对服务器实时负载情况,主要考虑 CPU 使用率 $U_c(i)$ 、内存使用率 $U_m(i)$ 和 IO 使用率 $U_d(i)$,分别对服务器性能的影响比例为 K_c 、 K_m 、 K_d (总值为 1)^[11]。则服务器的负载消耗权重 $W_l(i)$ 和剩余负载率 $W_r(i)$ 分别为:

$$W_l(i) = (K_c \times U_c i + K_m \times U_m i + K_d \times U_d i) \times T \quad (2)$$

$$W_r(i) = (1 - W_l(i)) \times W_i \quad (3)$$

再将连接数因素考虑进来,计算当前连接数权值 $W_n(i)$ 。计算此时集群中的各个服务器的连接数目 $N_i = \{N_1, N_2, \dots, N_n\}$,平均连接数 $\text{avgN} = (N_1 + N_2 + \dots + N_n) / n$ 。则每个服务器节点的连接数负载权值为 $W_n(i) = N_i / \text{avgN}$ 。

综合考虑服务器实际硬件使用情况和正在处理的

连接数,动态适应的调节负载权值,最终的实时负载权值为 $W_e(i)$,计算公式如下:

$$W_e(i) = W(i) - W_l(i) - W_n(i) = (1 - W_l(i) - W_n(i)) \times W_i \quad (4)$$

3 测试分析

测试实验中主要考虑的因素为并发连接数和响应时间^[12]。通过在不同量级连接数下,分别使用 least_conn 最少连接数法和基于连接数法经过优化过的自适应算法,测试对于响应的处理时间,来衡量优化过的算法是否具有实际效果^[13]。测试工具选用 Httpperf 和 webbench,Httpperf 能够根据不同的负载,记录性能结果并输出在终端。而 webbench 是一款著名的网站压力检测软件,能测试处在相同硬件上不同服务的性能以及不同硬件上同一个服务的运行状况^[14]。通过使用 webbench 分别模拟 300、500、700 和 1 000 个并发连接下,两种算法的实际事件响应时间,结果如表 1 所示;同时表 2 记录了每次实验时实际机器处理的数目。

表 1 算法的实际事件响应时间

并发数	least_conn 响应时间/ms	自适应算法响应时间/ms
300	27	31
500	48	44
700	167	136
1 000	345	238

表 2 每次实验时实际机器处理的数目

发起连接数	least_conn 实际处理数	自适应算法实际处理数
300	293	300
500	475	493
700	496	671
1 000	504	702

由表 1 可以看出,当并发数较低时(如 300),Nginx 自带的最少连接算法并不比文中优化过的算法慢,但是随着连接数逐渐升高到 500、700 甚至 1 000,经过优化过的动态算法,由于具有自适应的调节能力,响应时间优于最少连接法。在表 2 中,least_conn 算法能够承受的连接将近 500,高于 500 时出现明显的处理丢失,而自适应算法在同样的机器配置下明显可以承受到 700 左右,表明经过优化的算法可以更好地面对高并发连接,满足算法设计要求。

4 结束语

面对高并发连接下的请求处理,负载均衡技术尤为关键,通过将请求均衡地分配给服务器集群从而发

挥整体集群的性能。在对 Nginx 的结构进行分析后,理解其作为反向代理服务的高效高稳定的优越性,进而对其自带的负载均衡算法的不足进行分析。考虑集群间负载可能出现不均衡的情况,设计出一种自适应的动态调节算法,对服务器的实时性能使用率进行判断,从而优化连接的上游分配,实现集群性能优化。

参考文献:

- [1] 凌质亿,刘哲星,曹 蕾.高并发环境下 Apache 与 Nginx 的 I/O 性能比较[J].计算机系统应用,2013,22(6):204-208.
- [2] 王 艳,陈卫卫.基于 Nginx 替代 Apache 在高并发 WEB 负载均衡系统中的应用[J].电子测试,2015,20(6):88-92.
- [3] 田纯青.利用 Nginx 实现基于 URI 的 Web 负载分配[J].现代计算机,2009,11(7):187-190.
- [4] SERRANO D,PATIOMARTNEZ M,JIMNEZPERIS R. An autonomic approach for replication of internet-based services [C]//Proceedings of the IEEE symposium on reliable distributed systems. Naples, Italy: IEEE, 2008: 54-57.
- [5] 张炜森,陈 涛,李 康. Nginx 高并发负载均衡原理与策略比较研究[J].工业控制计算机,2018,31(1):85-86.
- [6] BITTAU A,BELAY A,MASHTIZADEH A. Hacking blind [C]//IEEE symposium on security and privacy. San Jose, CA, USA: IEEE, 2014: 154-160.
- [7] BRCIC P. Improving the performance of physical servers using a proxy servers accelerators [C]//21st telecommunications forum telfor. Belgrade, Serbia: IEEE, 2013: 43-48.
- [8] 张 尧.基于 Nginx 高并发 Web 服务器的改进与实现[D].长春:吉林大学,2016.
- [9] 杨 薇,赵 亮.Web 服务器性能优化研究[J].电子技术与软件工程,2016(13):20.
- [10] LIU Chaoping,LI Feng. The design and implementation of exquisite course website [C]//International symposium on information technologies in medicine and education. Hokkaido, Japan: IEEE, 2012: 43-46.
- [11] 李 彬,朱亚兴. Nginx 在实现网站负载均衡方面的研究[J].信息与电脑:理论版,2013,25(11):49-50.
- [12] 陶 辉.深入理解 Nginx [M].北京:机械工业出版社,2016.
- [13] 赵峡策.基于 Nginx 和 Memcache 的负载均衡集群架构设计[J].电子技术与软件工程,2014,23(5):39-40.
- [14] CHI X,LIU B,NIU Q. Web load balance and cache optimization design based nginx under high-concurrency environment [C]//2012 third international conference on digital manufacturing and automation. [s. l.]: [s. n.], 2012: 62-67.