计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(6):243−246 [doi: 10.15888/j.cnki.csa.006920]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

LVS 资源负载策略应用研究①

李姗姗[1,2], 高　岑[2], 王美吉[2], 李冬梅[2], 焦艳菲[3]

[1](中国科学院大学 计算机控制与工程学院, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)
[3](沈阳高精数控智能技术股份有限公司, 沈阳 110168)
通讯作者: 李姗姗, E-mail: 838329423@qq.com

摘　要: LVS (Linux Virtual Server) 是提高云平台资源利用率的方案之一, 但是由于 LVS 负载均衡算法的权值设置不科学以及在分配连接请求时不能实时地平衡任务, 导致云环境中的服务器负载失衡, 降低了系统对外提供服务的能力.针对以上问题, 本文将模拟退火算法和加权最小连接算法相结合, 提出一个基于最佳负载因子的负载均衡策略.并通过实验证明, 最佳负载因子策略能够使集群中节点负载更加均衡, 极大程度上提高了集群资源利用率.

关键词: 负载均衡; LVS; WLC; 集群系统; 模拟退火

引用格式: 李姗姗,高岑,王美吉,李冬梅,焦艳菲.LVS 资源负载策略应用研究.计算机系统应用,2019,28(6):243–246. http://www.c-s-a.org.cn/1003-3254/6920.html

Application Research of LVS Resource Load Strategy

LI Shan-Shan[1,2], GAO Cen[2], WANG Mei-Ji[2], LI Dong-Mei[2], JIAO Yan-Fei[3]

[1](School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110168, China)
[3](Shenyang Golding NC Technology Co. Ltd., Shenyang 110168, China)

Abstract: Linux Virtual Server (LVS) is one of the solutions to improve the resource utilization of cloud platform. However, because the weight setting of the LVS load balance algorithm is unscientific and the task cannot be balanced in real time when the connection request is allocated, the server load in the cloud environment is unbalanced, which reduces the ability of the system to provide external services. In view of the above problems, the simulated annealing algorithm and weighted least-connection algorithm are combined in this study to offer an improved balance strategy with optimal load factor. Experiments show that the optimal load factor strategy can make the node load in the cluster more balanced, so that the high availability of the cloud platform is improved.

Key words: load balancing; LVS; WLC; cluster; simulate anneal

1　引言

LVS 能够很有效地提高云平台的高可用性, 其基于 IP 技术和内容请求分发的集群负载方案, 能够把多台服务器构成一个高可用、高可伸缩和高性能的虚拟服务器集群系统. 由于其开源优势和出色的稳定性, LVS 受到了主流市场青睐. 但随着日益增长的多样化的海量数据请求以及网络请求中的高并发性, LVS 的调度策略逐渐无法满足服务需求, 改进原来的策略来提升 LVS 的负载均衡[1,2]能力, 为网络用户提供更高质量的服务成为当前研究的一个热点.

负载均衡一直以来都是服务器集群一个热门话题, 为了提高集群的资源利用率, 许多新策略和解决方案不断地被提出, 已经由最初的只能进行静态调度资源到可以根据系统运作情况进行动态资源分配, 再到后

① 收稿时间: 2018-11-08; 修改时间: 2018-12-04, 2019-01-03; 采用时间: 2019-01-07; csa 在线出版时间: 2019-05-25

来的自适应分配. 对此, 已经有很多国内外相关的技术人员在该方面做出了突出的贡献. 例如, IBM 公司提出的 Web Sphere 相关的一套 Web 服务器, 其提供了有效的集群解决方案和优秀的负载均衡能力. Microsoft 公司提出了具有高可用、高可伸缩性的网络负载技术 (NLB)[1]以及组件负载均衡技术 (CLB)[2]. 在国内, 诸多高校也致力于研究负载均衡策略。例如, 国防科技大学章文嵩博士支持开发的 Linux 虚拟服务器项目, 清华大学研发的可伸展的 Web 服务器集群系统.

2 LVS 架构及原理

2.1 LVS 结构

LVS[3]主要有负载调度器、服务器池、共享存储组成. 它们通过高速 LAN 相互连接, 构成 IPVS. 当系统升级时, 可以避免集群瘫痪.

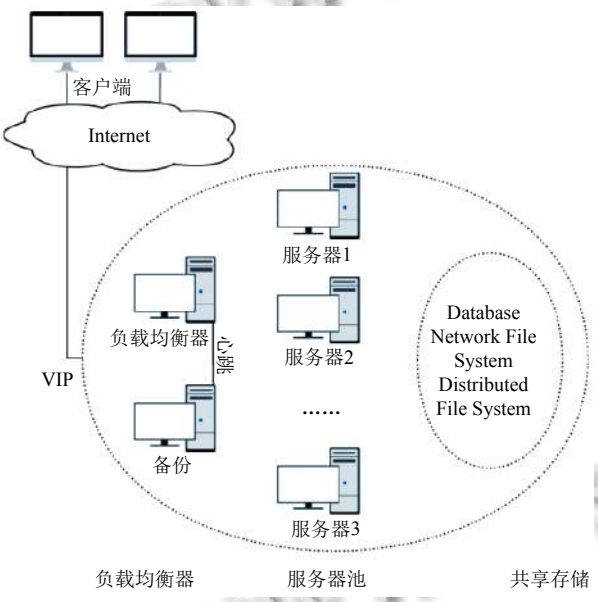

图 1 LVS 集群架构

2.2 LVS 原有的调度算法

LVS 有十种负载调度算法, 其中, 静态调度算法不考虑服务器中负载状况, 动态调度算法能根据负载的变化动态的调整请求调度策略.

静态调度算法: 轮叫调度算法、加权轮叫调度算法、源地址散列算法、目标地址散列算法.

动态调度算法: 最小连接数调度算法、加权最小连接调度算法、最短延迟调度调度算法、永不排队调度算法、基于局部性的最小连接调度算法、带复制的基于局部性最小连接调度算法.

3 基于最佳负载因子的调度策略

加权最小连接算法 (WLC) 是目前负载效果较好, 应用极为广泛的算法. 所以根据 WLC 的特性, 进行策略的改进.

3.1 加权最小连接算法 (WLC)

WLC[4,5]算法根据 RS 的处理能力, 为 RS 添加一个权值W_i来代表 RS 的处理能力. 当有新请求到达时, 会从分派链表中找出连接数量与权重比值最小, 且权重不为 0 的 RS. 若找到满足要求的 RS, 则返回该 RS 的指针; 否则返回空指针.

假设有一个集群$s=\{s_0,s_1,\cdots,s_{n-1}\}$, $C(S_i)$代表节点S_i的连接数, $W(S_i)$代表节点S_i的权重. 所有服务器当前的总连接数为$C_i=\sum C(S_i)$ $(i=0,1,\cdots,n-1)$. 当前新的任务被分配给节点S_m, 只有当节点$C(S_m)\times W(S_i)<C(S_i)\times W(S_m)$.

3.2 WLC 算法的影响因素

(1) 节点性能主要是与 CPU、内存、I/O 设备、外部存储等状况有关. 目前扩充外部存储器的容量较易实现, 而 I/O 设备和网络的带宽受制于客观条件; 如果太多的这些信息计算权值, 则节点的额外开销会过多.

(2) 权值设置的科学性. 服务器的权值是工作人员靠经验大体估算出来的, 权值设置不能很好地反映服务器的性能.

(3) 连接数不能准确反映出服务器资源的真实使用情况, 它只代表当前服务器的请求访问数量, 不同任务对资源占用情况也不尽相同.

3.3 最佳负载因子调度策略

针对原始的负载均衡算法中人为设置的权值, 负载因子越多, 其对应的权值则越多, 则计算结果与实际情况产生的偏差越大的情况. 在改进的算法中, 首先采取模拟退火算法[6,7]选择一个最佳的负载因子来表示节点的实际负载状态, 反应节点的实际性能.

对于 LVS 服务器来说, 为了能达到负载均衡, 首先选取 CPU 使用率、内存使用率和宽带使用率动态负载因子. 通过实时采集节点的信息, 选取建立节点资源模型和任务耗能模型.

节点资源模型$X_i=[X_{\text{cpu}_i},X_{\text{mem}_i},X_{\text{net}_i}]$, 其中, X_i表示第i个服务器可用的资源量, X_{cpu_i}表示 CPU 能力, X_{mem_i}表示内存容量大小, X_{net_i}表示带宽大小; 任务耗能

模型$Y_i=[Y_{\text{cpu}_i},Y_{\text{mem}_i},Y_{\text{net}_i}]$, 其中, Y_i表示第i个服务器单位时间内能耗量, Y_{cpu_i}表示 CPU 执行数量, Y_{mem_i}表示内存占用量, Y_{net_i}表示网络数据传输量.

各个指标的使用率如下: CPU 使用率为$U_{\text{cpu}}=X_{\text{cpu}}/Y_{\text{cpu}}$, 内存使用率为$U_{\text{mem}}=X_{\text{mem}}/Y_{\text{mem}}$, 宽带使用率$U_{\text{net}}=X_{\text{net}}/Y_{\text{net}}$.

计算各因子的权值向量:

$$\text{CPU权值向量为}\alpha_{\text{cpu}}=\left|\frac{\text{cov}(U_{\text{cpu}},Y)}{\sqrt{D(U_{\text{cpu}})}*\sqrt{D(Y)}}\right|$$

$$\text{内存权值向量为}\alpha_{\text{mem}}=\left|\frac{\text{cov}(U_{\text{mem}},Y)}{\sqrt{D(U_{\text{mem}})}*\sqrt{D(Y)}}\right|$$

$$\text{宽带权值向量为}\alpha_{\text{net}}=\left|\frac{\text{cov}(U_{\text{net}},Y)}{\sqrt{D(U_{\text{net}})}*\sqrt{D(Y)}}\right|$$

算法思想: 通过负载均衡器收集的节点信息, 采用模拟退火算法选择一个最佳负载因子. 模拟退火算法可以较好的解决组合优化问题, 并且最终结果与初始状态无关, 本文选用该算法来确定具体的权值向量. 将结果放进权值向量组:

$\alpha_1=(\alpha_{\text{cpu}_1},\alpha_{\text{cpu}_2},\cdots,\alpha_{\text{cpu}_n})$, $\alpha_2=(\alpha_{\text{mem}_1},\alpha_{\text{mem}_2},\cdots,\alpha_{\text{mem}_n})$, $\alpha_3=(\alpha_{\text{net}_1},\alpha_{\text{net}_2},\cdots,\alpha_{\text{net}_n})$, $\alpha'=(\alpha_1{}^T,\alpha_2{}^T,\alpha_3{}^T)$, 定义$U_i=(U_{\text{cpu}_i},U_{\text{mem}_i},\cdots,U_{\text{net}_i})$, 具体的服务器负载为$L_i=U_i*\alpha'$; 定义模型$T=(L,R,S)$, 其中:$T$为响应时间, L为负载情况, R为网络请求, S为选择函数来描述负载和请求的映射关系, 目标函数定义为$O=\overline{\sum T}$. 根据模拟退火算法:$S=SA(\alpha',T,O)$收敛时, 目标函数得到最小值. 算法的迭代过程:

Step1. 对现有解集α'中的元素进行替换, 产生新解;

Step2. 将新解带入目标函数, 计算新的目标函数O;

Step3. 以$\exp\left[\frac{O(i)-O(i-1)}{t}\right]$的概率接受新解, 其中$t$为平均响应时间, $O(i)\geqslant O(i-1)$;

Step4. 降温, 确定t<0.1 为算法终止条件, 停止后输出的最优解α即为最佳负载因子的权值向量.

通过收集到的节点信息, 计算每个服务器的最佳负载因子α相对应的负载率, 并给他们设置一个中间阈值, 负载率大于等于 80% 为欲过载区; 负载率小于 80% 为适用区. 根据任务选择适用区, 然后在适用区内选用经典 WLC[8–10]算法选出一个最合适的节点.

4 实验验证

4.1 实验方案

根据本文改进的策略, 结合实验室已搭建好的 OpenStack 平台进行验证, 并且资源配置如表 1.

表 1 云主机的资源配置

项目名称	内存 (GB)	硬盘 (GB)	数量
VM1	2	20	10
VM2	4	20	10
VM3	8	20	10
VM4	10	20	10
VLM	2	50	4

4.2 实验过程

负载均衡算法优劣一般是以集群的平均响应时间及服务器实时负载量作为评价指标. 响应时间越短、负载量越稳定, 说明集群的负载能力越好. 为了验证算法的有效性, 本文选用 WAS 压力测试软件, 在以下两种场景中对 Linux 服务器集群进行实验验证.

(1) 在测试集群瞬时压力的场景中, 设置多梯度的不同并发量向集群系统发送一定数量的请求数据包, 并且将其平均响应时间记录下来. 本文选取十个不同梯度的请求量 (单位: 个) 为:500、1000、1500、2000、2500、3000、3500、4000、4500、5000.

(2) 在测试持续高负载的场景中, 使用 WAS 软件模拟用户持续向集群系统发送一定数量的请求连接, 测试在该时间段内是否进行负载调整保证真实服务器的相对稳定. 观察并记录服务器请求成功的情况. 本文设置持续请求的时间为 120 s.

4.3 实验结果及分析

本文采用改进后的最佳负载因子策略与传统的 WRR 算法和 WLC 算法进行实验对比, 从集群角度分析, 响应时间越短、服务器负载量越高, 说明系统负载能力越强, 在相同时间内接受和处理的请求就越多.

图 2 测试结果显示, 当集群请求量数量较少时, 本文改进的策略和传统的策略平均响应时间相差不是很大. 但是随着请求量的增多、并发量的增大, 本文改进策略的平均响应时间要短很多. 当请求量达到 5000 个时, 本文策略的响应时间要比 WRR 算法和 WLC 算法分别少约 3.7 s、7.1 s, 很大程度上优化了集群系统对资源请求的处理速度, 节省了响应时间, 大大提高了系统资源的利用率.

图 3 测试结果显示, 本文改进的算法, 能够使得服务器的在一段时间内保持实时负载率在[57%, 73%]之间, 而其他两种传统策略的实时负载率分别在[50%, 85%]、[55%, 82%]之间. 由此可以得出, 在相同一段时间内, 本文改进策略更能使服务器的整体负载处于一个稳定状态, 可以将请求均衡的分配到各个服务器上,

几乎没有出现某台服务器负载倾斜的情况.

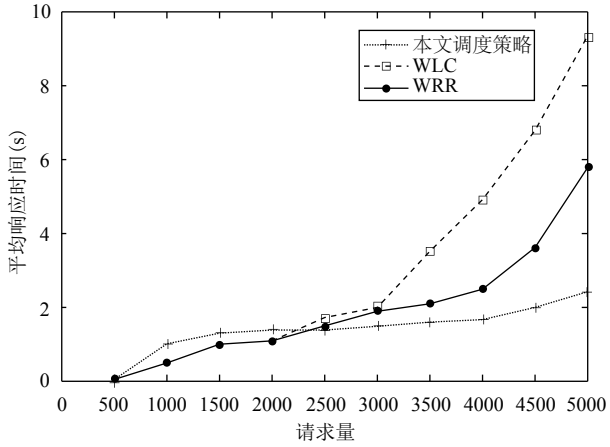

图 2 平均响应时间对比图

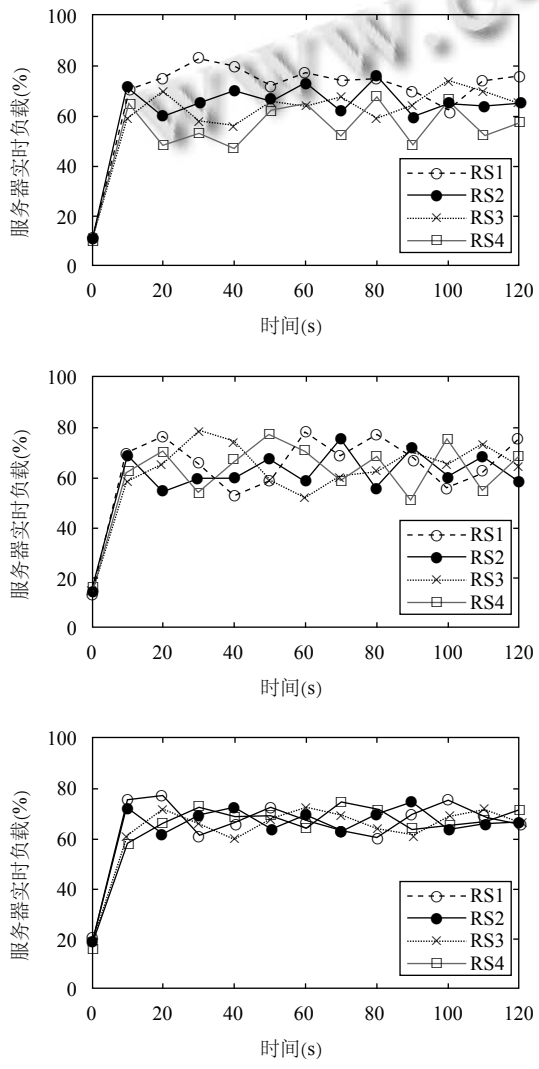

图 3 服务器负载情况对比

由以上结果可以得出, 在高并发、数据请求量巨大的集群系统中, 本算法在降低系统平均响应时间和均衡各服务器节点性能上具有很大的优势.

5 结语

本文通过分析现有的算法的优点与不足, 针对集群的负载因子越多对应权值越多而导致负载偏差的问题, 设计出一种将模拟退火算法和 WLC 算法相结合的最佳因子负载策略. 通过实验验证了本文改进的策略能够使得集群中的节点负载更加均衡, 很大程度的提高了集群资源的利用率. 该策略对于避免云平台中服务器负载倾斜、提高其负载能力有着很大的使用价值, 针对集群系统的高并发性、及负载量巨大的问题有着十分重要的意义. 但是本文在集群系统出现负载倾斜时如何将系统快速的调节到均衡的状态等方面的研究仍有不足, 有待进一步研究和优化本算法.

参考文献

1 刘琨. 云计算负载均衡策略的研究[博士学位论文]. 长春: 吉林大学, 2016.

2 蒋维成, 李兰英. 云计算资源动态配置策略. 电脑知识与技术, 2016, 12(26): 42–43, 66.

3 常兴磊. 基于 LVS 集群的一种动态负载均衡算法的研究与实现[硕士学位论文]. 长沙: 湖南师范大学, 2015.

4 罗南超. 云计算下均衡负载的差异性资源调度算法优化. 科学技术与工程, 2017, 17(34): 86–91. [doi: 10.3969/j.issn.1671-1815.2017.34.014]

5 王志刚, 常兴磊, 胥茜. LVS 集群的一种动态负载均衡方法. 福建电脑, 2017, 33(10): 111–113.

6 姚壹壹, 王玲鹏, 金科扬, 等. 基于模拟退火算法最优物流配送问题的应用. 宁波工程学院学报, 2018, 30(1): 39–44. [doi: 10.3969/j.issn.1008-7109.2018.01.007]

7 蒋美云. 基于模拟退火算法优化的 BP 神经网络预测模型. 软件工程, 2018, 21(7): 36–38.

8 Gayathri G, Latha R. Implementing a fault tolerance enabled load balancing algorithm in the cloud computing environment. International Journal of Engineering Development and Research, 2017, 5(1): 249–256.

9 Zhu P, Zhang JX. Load balancing algorithm for web server based on weighted minimal connections. Journal of Web Systems and Applications, 2017, 1(1): 1–8.

10 Kaur K, Kaur A. A hybrid approach of load balancing through VMs using ACO, MinMax and genetic algorithm. International Conference on Next Generation Computing Technologies. Dehradun, India. 2017. 615–620. [doi: 10.1109/NGCT.2016.7877486]