

一种改进的 Linux 集群系统负载均衡算法

涂俊英

(孝感学院 计算机与信息科学学院, 湖北 孝感 432000)

摘 要: 提出一种基于改进的 Linux 集群系统负载均衡算法. 首先通过修改负载权值计算指标, 然后通过动态反馈机制收集各个服务器的实时负载, 负载均衡器根据收集的负载信息动态调整服务器权值, 最后根据权值选择最优服务器去处理新的任务. 仿真测试结果表明, 改进负载均衡算法提高了整个系统的响应能力和吞吐量, 较大幅度地提高系统的综合性能.

关键词: 虚拟服务器; 集群系统; 负载均衡; 调度器

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2012)03-0106-04

An Improved Load Balancing of Linux Cluster System of Linux

TU Jun-ying

(School of Computer and Information Science, Xiaogan University, Xiaogan 43200, China)

Abstract: In view of the current load balancing algorithm shortcomings, an improved Linux cluster system load balancing algorithm is proposed based on load balancing mechanism. Firstly, the load weight indexes are modified, and the real time server load is collected by the dynamic feedback mechanism, the weights are dynamically adjusted according to the load information of servers collected by the load balancer, finally the optimal server is selected according to the weights to handle the task. The simulation test results show that the improved load balancing algorithm has improved the overall system response capacity and throughput, if greatly improves the comprehensive performance of the system.

Key words: Linux Virtual Server; cluster system; load balancing; scheduler

1 引言

服务器集群技术(cluster server)其具有高可靠性、扩展性好, 价格比超级计算机低廉, 强大的计算能力, 其中 Linux 虚拟服务器集群(LVS)是大规模数据处理的主要方案^[1]. 负载均衡是虚拟服务器集群系统中的核心技术. 负载均衡技术按照均衡策略可分为静态负载均衡和动态负载均衡两种, 而决定负载均衡核心是负载均衡算法. 目前常用的 LVS 负载均衡算法主要有: 轮转法、加权轮法、连接法、加权最少连接法、目标地址散列调度算法和源地址散列调度算法^[2-3]. 其中每种算法均存在各自的不足, 其中 LVS 系统中性能最优的算法为加权最少连接算法, 其依据的连接数, 考虑了服务器中各台性能是有

差异的, 但是不能动态反映服务器的实时负载情况, 有待进一步改进和研究^[4].

针对加权最少连接算法存在的不足, 为更好的使 Linux 集群系统负载均衡, 本文在加权最少连接算法基础上, 引入反馈机制对其进行改进. 最后通过仿真对比实验对改进后的加权最少连接算法性能进行验证.

2 LVS 的通用体系结构

LVS 群集系统通过采用内容请求分发技术和 P 负载均衡技术, 将用户的请求均衡、合理的分配到不同的服务器上, 提高系统的吞吐量, 从而将一组服务器集群成一个高可靠性、高性能、扩展性强、易管理的负载均衡系统. LVS 群集系统一般采用三层

结构,其主要包括负载调度器(load balancing scheduler)、服务器池(Server pool)和共享存储(shared storage),这几个部分通过高速网络连接,很好的防止了系统规模增大时互连网络成为整个系统的瓶颈^[5]. LVS的通用体系结构具体如图1所示.

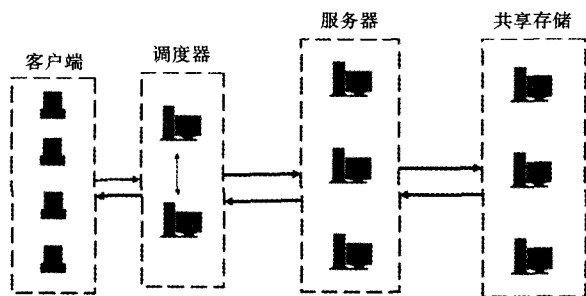


图1 LVS的通用体系结构

3 传统加权最少连接算法

加权最少连接算法(weighted least connections scheduling, WLC)是对最少连接算法的改进,是一种动态的负载均衡算法^[6]. WLC是依据当前网络服务器上的连接数作为选择的依据,每一个服务器上的连接数记录在负载均衡器中,其根据每一个服务器的性能,给其分配一个权重,相对于最小连接算法, WLC加入考虑实际服务器处理能力的因素,服务器的缺省权值为1,服务器的权值可以由系统管理员动态地设置. WLC负载调的基本思想为:当有用户进行分配请求时, Linux 集群系统的负载均衡器会优先考虑具有较大权重值的节点,尽量保证服务器的连接数和其权值成比例,当节点出现故障时,其权值设置为0,不给予分配负载.

设有一组服务器 $S = \{S_0, S_1, \dots, S_{n-1}\}$, 服务器 S_i 的权值为 $W(S_i)$, $Con(S_k)$ 表示第 k 个服务器 S_k 当前的连接数, i 表示上一次选择的服务器, 根据 WLC 算法思想可知, 当有连接请求达到时, 负载均衡器会将请求分配给满足条件的 S_j , 即有:

$$\frac{Con(S_j)}{W(S_j)} = \min \left\{ \frac{Con(S_k)}{W(S_k)} \right\} \quad (1)$$

其中, $W(S_j) > 0$, 且 $k=0, 1, \dots, n-2, n-1$.

由于 Linux 系统中不允许出现浮点除法运算, 且除法比乘法要用的 CPU 周期要长, 因此为了避免除法运行的出现, 可以对式(1)进行如下优化:

$$Con(S_j) * W(S_k) > Con(S_k) * W(S_j) \quad (2)$$

算法具体执行过程为:

for ($j=0; j < n; j++$)

```
{
    if ( $W(S_j) > 0$ )
    {
        for ( $k=j+1; k < n; k++$ )
        {
            if  $Con(S_j) * W(S_k) > Con(S_k) * W(S_j)$ 
                 $k=j$ ;
        }
        return  $S_j$ ;
    }
}
return NULL
```

综合上述可知, 加权最少连接法的流程如图2所示.

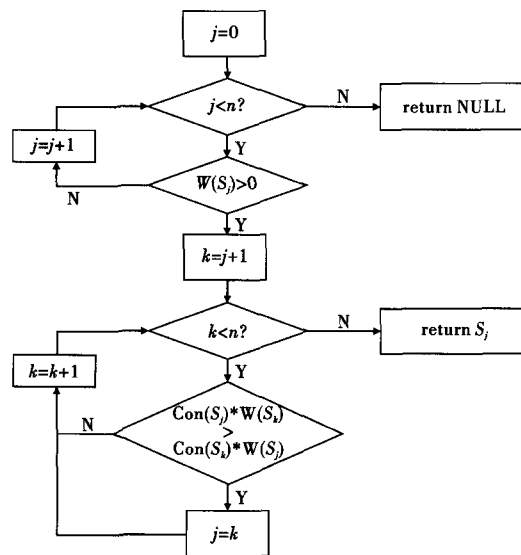


图2 WLC的工作流程

WLC算法相对于最少连接算法, 负载均衡调度并行性提、有效性有了较大的改善, Linux 集群系统性能也相应得到了提高, 避免了最少连接法的不稳定缺陷. 然而 WLC 还存在许多不足, 具体如下:

(1) 各台服务器的权值均凭管理员经验进行设置, 具有较强的主观性和盲目性, 导致权重值不能很好准确反映服务器的处理能力.

(2) 负载调度服务器权值是静态固定的, 不能进行修改.

(3) 仅采用连接数衡量一个服务器负载均衡状况不太准确, 因此连接数并不能代表负载状况.

为了使 Linux 集群系统负载更加均衡, 本文引入反馈机制, 对 WLC 的不足之处进行相应的改进, 使改进后的负载均衡调度算法能够更加准确的反映每一个服务器的实时负载状态, 最大限度的发挥各

服务器性能。

4 动态负载均衡算法

4.1 基于反馈机制的动态负载均衡模型

设 Linux 集群系统包含服务器为 $S=\{S_i\}, i=0,1,2,\dots,n-1$, $W(S_i)$ 表示服务器 S_i 在反馈调整前的处理能力权值, $Load(S_i)$ 表示此时的服务器负载情况, 即 CPU 的利用率、内存使用率等信息。那么引入反馈机制的动态负载均衡模型如图 3 所示。

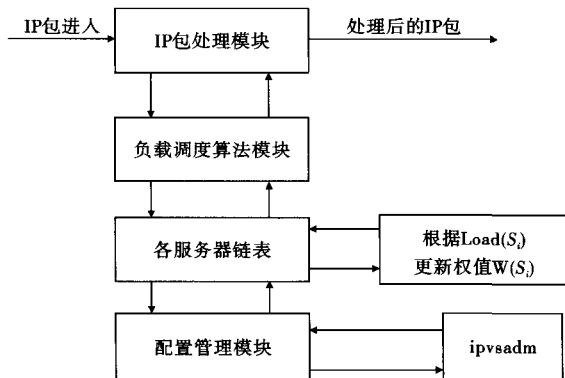


图 3 动态负载均衡模型

4.2 服务器负载状态权值的指标选择

由于基于反馈机制的动态负载均衡算法的权值能够真实的反映真实服务器的负载状, 由于每一类的集群系统面向的应用不同, 本文的集群系统是面对 Web 服务的, 因此选择 CPU 利用率、内存使用率和响应时间作为服务器负载状态权值的评价指标。当负载均衡器给每台服务器发送信息后, 每台服务器会对该信息进行回应, 返回的 $Load(S_i)$ 就是反映真实服务器的 CPU 利用率、内存使用率和响应时间的一个权值。

4.3 服务器负载权值的计算

在 Linux 集群系统中, CPU 利用率、内存使用率和响应时间反映服务器负载状态的重要程度不一样, 因此需要对它们设置相应的系数, 系统越大表示越重要。具体计算公为:

$$Load(S_i) = R_1 * CPU(i) + R_2 * MEM(i) + R_3 * RESP(i) \quad (3)$$

其中, $R_1 + R_2 + R_3 = 1$, $CPU(i)$ 表示 CPU 利用率, $MEM(i)$ 表示内存使用率, $RESP(i)$ 表示响应时间。

4.4 动态负载均衡算法描述

动态负载均衡算法的具体描述如下:

(1) 负载均衡器周期性的向真实服务器发送请求信息, 想获得真实服务器的负载状态。

(2) 真实服务器向负载均衡器发送负载状态信息, 如果其综合负载值大于预先设定好的阈值, 那么就调整服务器负载状态权值: $W(S_i)' = 0$, 否则就与服务响应时间 $RESP(i)$ 计算整个的负载值 $Load(S_i)$ 。

(3) 计算服务响应时间 $RESP(i)$ 。负载均衡器向真实服务器发送请求信息, $TIME_1$ 表示发送时间, $TIME_2$ 表示收到响应信息时间, 那么服务器响应时间为: $TIME_1 - TIME_2$, $RESP(i)$ 越小就表示响应时间越长。

(4) 负载均衡器将 $RESP(i)$ 和综合负载值 L_i (partly) 合并组成完整负载值 $Load(S_i)$, 如果 $Load(S_i)$ 大于预先设定好的阈值, 那么 $W(S_i)' = 0$, 否则, 取观原始 $W(S_i)$ 的值, 通过采用相应的算法与 $Load(S_i)$ 计算出新的权值 $W(S_i)'$ 。

(5) 负载均衡器根据 $W(S_i)'$, 采用加权最少连接算法选择最优服务器, 并将用户请求分配给最优服务器。

因此动态负载均衡算法的工作流程如图 4 所示。

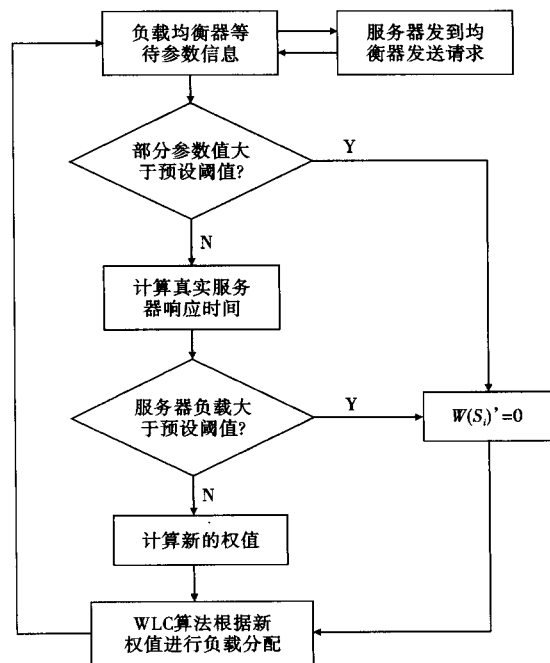


图 4 动态负载均衡算法的工作流程

5 仿真实验

5.1 仿真环境

为了验证本文提出的动态负载均衡算法的有效性和优越性, 采用 LVS-DR 形式搭建测试服务集群, 其有两台真实服务器、两台测试服务器和一

台负载均衡器,通过测试服务器对用户访问行为进行模拟,具体如图5所示.测试工具采用 Microsoft Web Application Stress Tool(WAS).

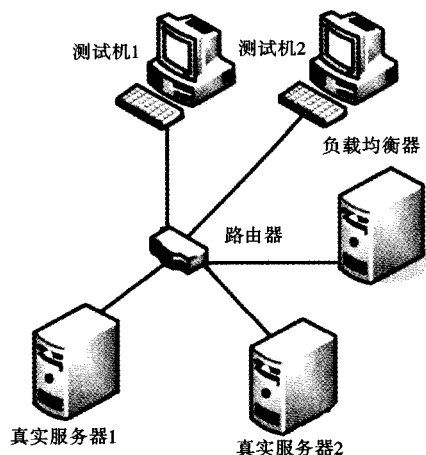


图5 LVS测试平台结构

5.2 算法性能分析

为了使本文算法的结果具有可比性,采用轮询算法(Round-Robin Scheduling, RRC)和WLC作为对比算法,采用平均响应时间(Time To First Byte, TTF)和集群系统的吞吐量 Bytes Recv Rate(in KB)作为评价指标。

5.2.1 平均响应时间对比

仿真实验表明基于本文算法、RRC、WLC三种算法集群系统的平均响应时间在连接数比较小时,三种算法的集群系统的平均响应时间相差不大,随着连接数不断增大时,本文算法的平均响应时间要小于RRC、WLC算法。

5.2.2 集群系统的吞吐量

比较本文算法、RRC、WLC三种算法集群系统的吞吐量,本文算法的集群系统吞吐量远远大于两种对比算法。结果表明本文算法大大改善了集群系统性能,能够更好为的用户提供服务,具有具有更

好的响应服务时间和更大的吞吐量,是一种有效的集群系统负载均衡,更加适合于群系统负载动态性变化。

6 结束语

本文对Linux下集群系统的负载均衡算法现状进行研究,引入负载反馈机制,提了一种改进的负载均衡算法。最后通过负载仿真测试对比实验对算法的性能进行验证,结果表明,相对于其它负载均衡算法,改进的负载均衡算法降低了系统平均响应时间,提高了系统的吞吐量,能够更好均衡Linux集群系统的各个节点负载,提高了系统的可靠性和稳定性,在集群系统负载优化中有着广泛的应用前景。

参考文献:

- [1] 杨爱民,刘韧,赵广华,等. 集群系统中基于MPI的并行GMRES(m)计算通信的研究及应用[J]. 微电子学与计算机,2009,28(9):127-130.
- [2] 高鸿峰,王一波,傅光轩. Netfilter 框架下IPv6防火墙的设计与实现[J]. 电子科技大学学报,2007,36(6):1427-14.
- [3] 郭成城,晏蒲柳. 一种异构Web服务器集群动态负载均衡算法[J]. 计算机学报,2005,28(2):180-183.
- [4] 刘健,徐磊. 基于动态反馈的负载均衡算法[J]. 计算机工程与科学,2003,25(5):65-68.
- [5] 刘东君,殷保群,郭东. 基于目标规划的服务器集群系统请求调度模型[J]. 计算机仿真,2010,27(12):100-103.
- [6] 郑伟民,石威,汪东升,等译. 高性能集群计算:结构与系统(第一卷)[M]. 北京:电子工业出版社,2001.

作者简介:

涂俊英 女,(1970—),副教授,硕士.研究方向为智能计算与网络安全。