基于云自适应遗传算法的 LVS 负载均衡改进算法①

郭 洪, 王监梁

(福州大学 数学与计算机科学学院, 福州 350108)

摘 要: 针对 LVS 系统存在负载倾斜问题, 改进了系统中带权值参数的负载均衡算法. 先通过服务器节点反馈的参数计算出服务器节点的负载, 再采用云自适应遗传算法根据负载为各服务器节点分配权值, 进而实现改进算法. 实验证明, 改进算法比传统算法提高约 10%的性能, 可使系统有效地实现动态负载均衡.

关键词: 负载均衡; Linux 虚拟服务器; 负载反馈; 服务器集群; 遗传算法; 云模型

Improved LVS Load Balancing Algorithm Based on Cloud Adaptive Genetic Algorithm

GUO Hong, WANG Jian-Liang

(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)

Abstract: Aimed at the problem of load skew in LVS system, this article improves load balancing algorithm with weight parameter of system. It first calculates server node's load according to parameters fedback by server node, then distributes weight to every server node according to its load by using cloud adaptive genetic algorithm, further realizes the improved algorithm. Experiment shows that, the improved algorithm is better about 10% than traditional algorithm in performance, and can make the system effectively realize dynamic load balancing.

Key words: load balancing; linux virtual server; load feedback; server cluster; genetic algorithm; cloud model

随着网络通信量的快速增长, 服务器负载增长迅猛导致经常超载, 特别是一些流行网站[1]. 为了解决服务器超载问题, 许多公司采用服务器集群来满足用户需求, 客户访问集群系统提供的网络服务就像访问一台高性能、高可用的服务器一样[2]. LVS(linux virtual server) 即为一种服务器集群系统, 它采用前台为负载均衡器和备用负载均衡器, 后台为多台服务器节点的模式, 负载均衡器根据调度算法分发请求给服务器节点. 但是传统调度算法在分发请求时未考虑服务器节点的负载以及系统整体的均衡性以致系统出现负载倾斜问题, 因此, 改进传统调度算法成为提高系统性能的关键.

一般的改进算法[3,4]采用根据服务器节点反馈的静态参数、CPU 利用率、内存利用率等参数确定服务器节点权值, 或者根据内容[5,6]和遗传算法[7,8]分发请求以提高系统性能, 但是都存在一定的缺陷. 如改进 WLC(weighted least-connection)算法[9]需要启动软件检测服务器节点在压缩或解压缩等功能上的性能, 消耗了服务器节点资源. 基于遗传算法的负载均衡算法[10]需要负载均衡器和服务器节点定期通信, 增加了网络压力. 为提高 LVS 系统性能, 本文改进了系统中带权值参数的负载均衡算法, 该改进算法根据服务器节点负载为服务器节点分配权值.

1 系统设计

改进算法的思想是在系统调用传统带权值参数的调度算法(WLC、WRR 等)运行时, 分别在负载均衡器和服务器节点上设置监视器(monitor)和检测器(detector), 如图 1 所示.

检测模块负责检测服务器节点参数信息, 并适时反馈服务器节点的参数. 监听模块负责发送信息至检测模块并获得各服务器节点的参数, 然后更新或保存

① 基金项目:国家自然科学基金(61075022)
收稿时间:2012-03-06;收到修改稿时间:2012-04-07

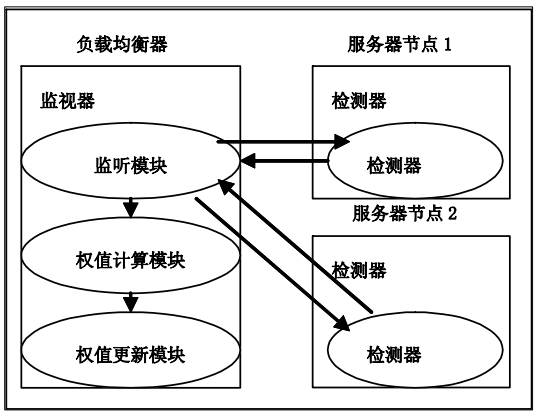

图 1 系统总体框架

数据结构中参数. 权值计算模块负责从数据结构中取出参数, 然后根据云自适应遗传算法为各服务器节点分配权值. 权值更新模块负责取出各服务器节点数据结构中的权值, 并且对服务器节点的权值进行更新.

监听模块先发送信息至所有检测模块, 检测模块收到信息便检测服务器节点动态和静态参数然后反馈, 当监听模块收到反馈信息后计算输入指标以及响应时间, 然后将上述四项参数存储至数据结构中, 权值计算模块取出数据结构中参数并利用云自适应遗传算法为各服务器节点分配权值并存储至数据结构中, 最后权值更新模块取出权值并更新各服务器节点的权值. 此后检测模块定期检测服务器节点使用情况, 当 CPU 利用率和内存利用率变化过大或系统总利用率过大时发送信息到监听模块, 监听模块收到信息后又发送信息至所有检测模块以获得参数, 如此循环.

权值计算模块分配权值时是根据参数计算出各服务器节点的负载, 然后利用遗传算法根据负载为各服务器节点分配权值. 由于遗传算法中的交叉概率 Pc 和变异概率 Pm 固定, 导致遗传算法容易出现过早收敛和局部最优现象, 因此, 本文利用云模型的随机性和趋向性特点, 并根据适应度值自动生成 P_c 和 P_m.

2 系统实现

2.1 检测模块

当检测模块首次收到监听模块发送的请求信息时, 检测模块通过系统文件获得 CPU 频率、内存大小和磁盘 I/O 速率等静态参数(static)以及 CPU 利用率和内存利用率等动态参数(auto), 然后将 static 和 auto 发送至 monitor.

$$static = a \times CPU_{fre} + b \times Mem + c \times Speed \quad (1)$$

$$auto = e \times CPU_{ratio} + f \times Mem_{ratio} \quad (2)$$

式中 CPU_{fre}、Mem、Speed 分别为 CPU 频率、内存大小、磁盘 I/O 速率, a、b、c[0,1]且和为 1. CPU_{ratio}、Mem_{ratio} 分别为 CPU 利用率和内存利用率, e、f[0,1]且和为 1. 此后检测模块定期检查 CPU 利用率变化率和内存利用率变化率以及系统总利用率.

$$Change_{cpu} = \left|CPU_{ratio1} - CPU_{ratio2}\right| / (t_2 - t_1) \quad (3)$$

$$Change_{Mem} = \left|Mem_{ratio1} - Mem_{ratio2}\right| / (t_2 - t_1) \quad (4)$$

$$Total_{ratio} = CPU_{ratio} + Mem_{ratio} \quad (5)$$

式中 $Change_{cpu}$、$Change_{Mem}$ 和 $Total_{ratio}$ 分别为 CPU 利用率变化率、内存利用率变化率和系统总利用率, CPU_{ratio1}、CPU_{ratio2} 分别为服务器节点在时刻 t_1 和 t_2 秒时的 CPU 利用率, Mem_{ratio1}、Mem_{ratio2} 分别为服务器节点在时刻 t_1 和 t_2 秒时的内存利用率. 当上述三项指标超过阈值时, 则发送信息至 monitor, 并在接收到 monitor 的请求信息后反馈服务器节点的 auto.

2.2 监听模块

监听模块首先发送请求信息至所有服务器节点, 获得所有服务器节点的 static、auto、响应时间、输入指标, 并将上述参数存储至数据结构中. 此后, 当监听模块接收到检测模块发送的信息时, 发送请求该服务器节点的 auto 请求信息, 并计算服务器节点的输入指标以及从 t 时刻发送请求信息至 t、时刻收到反馈信息所耗的响应时间 T(T=t’-t), 最后, 更新数据结构中的参数. 服务器 Si 的输入指标 INPUTi 为其新连接数与 n 台服务器节点收到平均连接数的比值[11], 数据结构中包含节点 IP、static、auto、INPUT、T、节点权值 W.

$$INPUT_i = \frac{N_i}{\sum_{m=1}^{n} N_m / n} \quad (6)$$

2.3 权值计算模块

本模块采用云自适应遗传算法计算各服务器节点的权值. 遗传算法借鉴生物进化论将要解决的问题模拟成生物进化的过程, 通过复制、交叉、突变等操作产生下一代的解, 并逐步淘汰掉适应度值低的解, 获得适应度值高的解. 经过多次这样的操作得出适应度

值很高的个体. 云模型是由李德毅教授提出的基于随机数学和模糊数学的理论, 采用数字特征(E_x,E_n,H_e)表示语言的数学性质. 期望 E_x 表示定性概念的中心, 熵 E_n 表示定性概念可度量的粒度, 超熵 H_e 表示定性概念样本发生的随机性. 云模型的随机性符合自然进化中基因突变的随机性, 趋向性符合遗传算法交叉与变异操作随着适应度值的增加需要逐渐减小发生交叉和变异的概率的特点. 因此, 通过云发生器产生交叉概率 P_c 和变异概率 P_m, 可以有效提高遗传算法收敛速度, 并且提高避免陷入局部最优的能力. 云自适应遗传算法流程如图 2 所示.

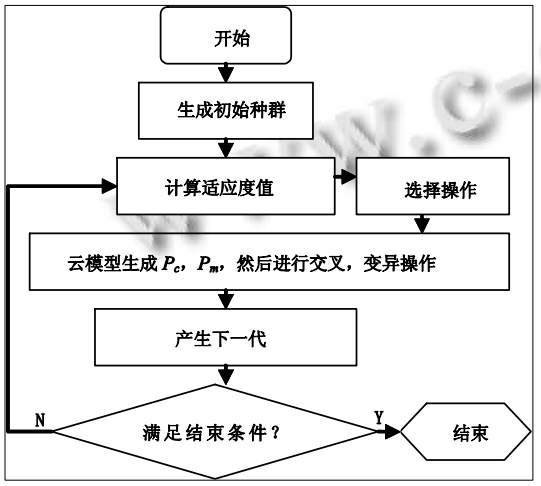

图 2　云自适应遗传算法流程

2.3.1 参数编码

云自适应遗传算法编码方式如图 3 所示, 图中的一行表格表示一个可行的权值分配方案. 方格中的参数为各服务器节点的权值.

图 3　参数编码

2.3.2 适应度函数

从数据结构中取出参数, 计算各服务器节点的负载状况 L_i(公式 7)和服务器节点负载 $Load_i$(公式 8):

$$L_i = \alpha \times auto + \beta \times INPUT_i + \gamma \times T \tag{7}$$

$$Load_i = L_i \times W_i / static \tag{8}$$

本文采用适应度函数(公式 9)表示系统整体的负载均衡状况. 适应度值越大表示系统负载均衡性越好, 否则相反, 适应度值 f∈[0,1].

$$f = 1 - \frac{\sqrt{\sum(\text{节点}i\text{负载}-\text{所有节点平均负载})^2}}{\text{负载总和}} \tag{9}$$

3.3.3 选择操作

选择操作采用常用的轮盘赌选择方法选择个体. 假设个体 i 的适应度值是 f (i),则选择概率为:

$$P(i) = \frac{f(i)}{\sum f(j)} \tag{10}$$

2.3.4 交叉与变异操作

交叉操作是模仿生物界把两条父染色体进行基因重组, 得到两条含有父代染色体部分基因的后代染色体. 变异操作是染色体上的基因按照一定的概率变异. 若交叉概率 P_c 和变异概率 Pm 设置不当易出现过早收敛或局部最优的现象. 因此, 根据个体的适应度值设置 P_c 和 P_m 比较合理, 当个体的适应度值比较大时, 减小 P_c 和 P_m 的值以减小破坏较优个体的机会; 当适应度值比较小时, 增加 P_c 和 P_m 的值以产生较优个体. 如采用二次方公式[12]确定 P_c 和 P_m 的值, 或根据云模型[13]的随机性和趋向性特点采用 X 条件正态云模型生成 P_c 和 P_m, 本文采用二维云模型生成 P_c 和 P_m. 基本二维云发生器实现如下:

Input:期望值(E_x,E_y),熵(E_{n1},E_{n2}), 超熵(H_{e1},H_{e2})

Output:多个三维点(x_i,y_i,z_i) ($0 \le i \le N$)

For(int i=1;i<=N;i++){

(x_i,y_i)=PCG(E_x,E_y,E_{n1},E_{n2});

(E_{n1i},E_{n2i})=PCG(E_{n1},E_{n2},H_{e1},H_{e2});

$$z_i = e^{-\frac{(x_i - E_{x1})^2}{2(E_{n1i})^2} - \frac{(y_i - E_{x2})^2}{2(E_{n2i})^2}}$$ }

根据二维云发生器生成 P_c、P_m 的算法如下:

$$E_{x1} = \overline{f} \tag{11}$$

$$E_{x2} = \overline{f} \tag{12}$$

$$E_{n1} = (f_{\max} - \overline{f}) / c_1 \tag{13}$$

$$E_{n2} = (f_{\max} - \overline{f}) / c_2 \tag{14}$$

$$H_{e1} = E_{n1} / d_1 \tag{15}$$

$$H_{e2} = E_{n2} / d_2 \tag{16}$$

$$(E_{n1}^{'}, E_{n2}^{'}) = PCG(E_{n1}, E_{n2}, H_{e1}, H_{e2}) \tag{17}$$

$$y = e^{-\frac{(f-E_{x1})^2}{2(E_{n1}')^2}-\frac{(f-E_{x2})^2}{2(E_{n2}')^2}} \tag{18}$$

$$P_c=\begin{cases}k_1\times y & f\ge \overline{f}\\ k_2 & f<\overline{f}\end{cases} \tag{19}$$

$$P_m=\begin{cases}k_3\times y & f\ge \overline{f}\\ k_4 & f<\overline{f}\end{cases} \tag{20}$$

式中 $\overline{f}$ 为平均适应度值，$f_{\max}$ 为最大适应度值，En1’、En2’为随机生成的值，f 为适应度值. 若种群规模和终止进化代数设置的较小则难以得到较优解，设置的过大会影响算法速率，因此本算法设种群规模为 30，终止进化代数为 25. 根据云模型“$3E_n$”规则，在论域上的绝大部分定量值出现在$[E_x-3E_n,E_x+3E_n]$区域上，若 E_n 值过小则使较优个体的 P_c 和 P_m 值比较大，易破坏较优解，通过实验证明当 $c_1=3$, $c_2=3.1$ 时比较合理. H_n 确定云滴的离散度，当值过小时云滴比较稠密影响随机性，值过大时云滴比较松散影响趋向性. 本文取 $d_1=10$, $d_2=11$. 由于 P_c 和 P_m 的值都与生成的 y 值成正比，因此本文采用公式(19),(20)表示 P_c、P_m.

2.4 权值更新模块

取出服务器节点数据结构中的权值，若新权值和设定的权值相同则不用更新，否则更新. 更新服务器节点权值有以下几种方法：一种方法是调用 setsockopt()函数将新权值传递到内核，另外一种方法是使用 ipvsadm 软件的 ipvsadm 命令修改权值，本文采用后者. 首先使用 sprintf()函数将命令 ipvsadm－e－t vip:port－r rip:port－g－w weight 转换为字符串形式，然后再由 system(*string)执行修改权值的命令.

3 系统测试

3.1 系统测试配置

将 monitor 和 detector 分别置于负载均衡器和服务器节点上. 系统测试需要在负载均衡器和服务器节点上安装 ipvsadm 和 heartbeat 软件，并采用直接路由方式配置 Web 集群系统，然后用 WAS(web application stress)软件测试系统性能，通过测试得出的算法性能，比较改进算法和原算法的性能. 测试配置如表 1 所示:

表 1 测试配置表

项目	*CPU*	*内存*	*操作系统*
服务器节点 1	双核 2.8G	2G	Entos5.6
服务器节点 2	双核 2.8G	2G	Entos5.6
负载均衡器	单核 2.8G	1G	Entos5.6
备用负载均衡器	单核 2.8G	1G	Entos5.6

3.2 系统测试结果分析

根据表 2 和表 3 得出当连接数比较小时，传统算法与改进算法性能相当,当连接数比较大时，改进算法性能明显比传统算法好,并且该改进算法比最小权值动态负载均衡算法[14](MW)性能要好. TTFB(time to first byte), Request per second(单位时间连接数).

表 2 算法的 TTFB(ms)比较

连接数	WLC	改进 WLC	WRR	改进 WRR	MW
100	276.43	289.96	287.15	299.33	294.57
200	682.89	682.42	800.42	798.53	795.66
300	1182.93	1038.06	1231.54	1156.41	1106.35
400	1584.11	1384.1	1615.15	1489.31	1499.63
500	1970.83	1722.67	2003.82	1802.21	1836.21
600	2894.25	2315.69	2945.21	2388.63	2456.68

表 3 算法的 Request per second(个)比较

连接数	WLC	改进 WLC	WRR	改进 WRR	MW
100	76.87	73.53	70.55	68.73	68.36
200	76.94	87.38	71.67	85.75	84.56
300	82.05	92	77.15	91.32	86.23
400	92.38	102.74	86.42	103.22	97.59
500	101.66	111.48	96.52	110.53	106.56
600	110.13	122.69	104.03	120.54	116.32

4 结语

该改进算法比较全面地考虑到如静态参数、动态参数、输入指标、响应时间等影响系统负载均衡的参数，克服了一般动态调度算法需要负载均衡器和服务器节点定期通信的缺点，减轻了网络压力. 并且利用云自适应遗传算法根据服务器节点的负载动态均衡地分配权值，有效地提高了算法性能并实现了系统负载均衡. 由于改进算法没有修改 LVS 系统内核，只要 LVS 系统中的调度算法涉及到权值，那么 monitor 和 detector 都可以移植其中.

参考文献

1 Xu Y, Xie XY. Research and Design on LVS Cluster System. Hu BG, Saguez C, Xie XY, Gomez C, ed. Proc of the 2009 IEEE International Workshop on Open-source Software for Sciectic Computer. Beijing: IEEE Press, 2009.68–72.

(下转第 115 页)

机系统包括智能控制层、数据库层、实时监控层、信息管理层. 下位机主要对各个需要监控的采集点进行部署, 采用 2 套西门子 S7-3 0 0 系列可编程控制器, 利用 P RO F I BUS 总线同上位机系统进行信息通信. 上位机系统由 1 台服务器和 4 台客户机组成, 其中 2 台客户机不作画面监控, 只做变量报表监控, 各上位机及服务器通过交换机连接组成一个局域网, 服务器上安装有 Wincc6.2 完全版和WinCC网络报表系统. 在设置好服务器IP地址和WinCC网络报表系统的连接参数后, 用户可以通过网络访问 WinCC 网络报表系统. 在获得授权登陆后, 用户可以根据自己的需求, 选择归档变量、归档时间和时间粒度等参数, 在递交参数后, 系统自动生成报表, 并提供EXCEL格式报表文件的下载.基于Ajax技术的WinCC网络报表系统高效、灵活地提供了打印日、月、年报表和观察趋势图的功能, 使得污水处理厂对各个处理工艺段的水质参数能进行横向和纵向对比, 为污水处理过程中的包括能耗计划、统计、对比、预测提供了依据.

在项目的实际使用中发现基于 Ajax 技术的 WinCC 网络报表系统有如下优势:

(1) 对归档数据的访问效率大大提高, 随着数据量的不断增大, 访问速度明显要比原来快很多.

(2) 对用户而言, 界面更加丰富, 控制更加便捷, 对数据的统计分析更加全面和透彻, 有利于用户做出正确决策.

6 小结

基于Ajax技术的WinCC网络报表系统结合了数据库技术、综合自动化信息处理技术和 Web 技术, 通过 ASP 程序将 WinCC 归档数据库的数据进行查询处理, 并使用 Ajax 技术显示在页面上. 系统具有灵活的报表显示、趋势图表的显示、打印报表和导出报表为 Excel 表的功能, 同时系统还具有很好的通用性和兼容性. 它基本满足了用户的要求, 有效的解决了报表的远程访问和数据共享的问题, 提高了数据处理的灵活性. 该系统已经在实际工程项目中得到应用, 效果良好.

参考文献

1 梁绵鑫,罗艳红,边春元,等.WinCC 基础及应用开发指南.北京:机械工业出版社,2009.

2 曹路圆,吴迪,刘征宇.基于 WinCC Web Navigator 的生产数据发布系统.制造业自动化,2010,(9):185−186.

3 姚文声.Ajax 技术在在线考试系统中的应用.电脑编程技巧与维护,2011,(16):69−70.

4 甄立东.西门子 WinCC V7 基础与应用.北京:机械工业出版社,2011.

5 赵秀梅.基于 WinCC 工控组态软件的关系数据库的研究.微型机与应用,2010,(6):1−2.

6 于晓云,刘健,温联山.基于 WinCC 软件的 Web 报表设计.工业控制计算机,2011,24(9):35−36.

(上接第 57 页)

2 Vrenios A.马朝晖,等译.Linux 集群体系结构.北京:机械工业出版社,2003.

3 田绍亮,左明,吴绍伟.一种改进的基于动态反馈的负载均衡算法.计算机工程与设计,2007,28(3):572−728.

4 郭全生,舒继武,毛希平,温冬蝉,郑纬民.基于 LVS 系统的负载动态平衡设计与实现.计算机研究与发展,2004,41(6):923−929.

5 梁彪,黄战.基于实时性能动态反馈的负载均衡算法.计算机系统应用,2010,19(3):183−186.

6 郑祺,周广平.基于内容分类的集群负载均衡算法.计算机系统应用,2011,20(5):47−50,74.

7 张维勇.Web 服务器集群的负载均衡中遗传算子的设计.计算机应用与软件,2010,27(4):209−211.

8 余燕芳,陆军.基于改进遗传算法的服务器端负载均衡算法.微电子学与计算机,2007,24(7):146−148.

9 Liu YY, Fang YK. Optimizing WLC scheduling algorithm of LVS.The 2010 International Conference on Computer Application and System Modeling, Chendu: IEEE Press, 2010,6:585−585.

10 维勇,张华忠,柳楠.基于遗传算法的服务器端负载均衡系统的设计.计算机工程,2005,31(20):121−123.

11 The linux Linux Virtual Server Project. Dynamic Feedback Load Balancing Scheduling. http://kb.linuxvirtualserver.org/wiki/Dynamic_Feedback_Load_Balancing_Scheduling. 2006 Aug.

12 申艳芬,董丽丽,张翔.基于改进遗传算法的给水管网最短路径求解.计算机仿真,2011,28(2):260−263.

13 戴朝华,朱云芳,陈维荣.云自适应遗传算法.控制理论与应用,2007,24(4):646−650.

14 Song SY, Xu JP, Cong Q. Dynamic Feedback Equalization Algorithm for Minimum Weight. In: Yi H, Wen DS, Sandhu PS, ed. The 2010 3rd IEEE International Conference on Computer Science and Information Technology. Beijing: IEEE Press, 2010.267−270.