

面向集群服务器大规模并发的改进负载均衡调度系统

杨 越, 闫连山, 张志勇, 李赛飞

(西南交通大学 信息光子与通信研究中心, 四川 成都 610031)

摘 要: 对常规的服务器集群架构进行了改进, 提出了决策器的概念, 并由决策器训练调度序列; 同时根据自适应小生境遗传算法提出了适应特征值作为适应值的评价标准, 并合理地应用于负载均衡调度系统中. 本文搭建网络环境模拟大并发测试, 实验数据与分析表明, 本方案的系统平均响应时间仅为 2 ms, 同时错误率趋近于 0, 相比改进前服务器更均衡地利用, 系统性能更稳定.

关键词: 负载均衡; 小生境遗传算法; 动态轮询

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2013)12-0054-03

A Modified Load Balancing System for Cluster Servers under Mass Concurrence

YANG Yue, YAN Lian-shan, ZHANG Zhi-yong, LI Sai-fei

(Center for Information Photonics & Communications, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: This article has made improvement in the architecture of regular server clusters, The concept of "decision maker" is put forward, which trains scheduling sequence; Moreover, according to adaptive niche genetic algorithm (ANGA), the adaptive eigenvalue is regarded as evaluation criterion which can be reasonably applied in the controlling system of load balance. Furthermore, a network environment is simulated by thousands of test. It is demonstrated by experimental data and analysis that the average responding time of the system is 2 ms. At the same time, error rate approaches 0. Compared with the system before improvement, the new system is more balanced use and stable in performance.

Key words: load balance; adaptive niche genetic algorithm (ANGA); dynamic round robin(drr)

1 引言

随着 Internet 的发展, 网络用户日益增加, 需要服务器集群系统有处理大量用户请求的能力. 目前, 服务器集群是解决服务器超载问题的最好解决办法. 服务器集群要求设计实现有效的请求机制和负载均衡策略, 要求能均衡地将系统的访问量分配给集群服务器中负载最少的节点. 目前已有的负载均衡算法, 一般分为静态调度和动态调度这两类算法^[1]. 动态调度算法在大多数情况下性能是优于静态调度算法的, 但它必须实时地收集、分析各个节点服务器的状态信息, 因此动态负载均衡算法会带来

额外的开销^[2].

基于以上情况, 本文主要以提高调度算法的适应性和减小算法开销为研究目的, 对负载调度算法及架构进行了研究, 提出将小生境遗传算法^[3]应用于负载均衡调度的问题上并修改遗传算子; 同时根据调度的实际应用问题改进了服务器集群架构.

2 算法概述

自适应小生境遗传算法 (adaptive niche genetic algorithm, ANGA) 可以避免在进化后期, 适应值高的个体大量繁殖, 满整个群体^[4]. 本文根据 ANGA 设计出了一种能适用于服务器集群的调度算法.

收稿日期: 2013-01-20; **修回日期:** 2013-03-19

基金项目: 铁道部重大项目(2012X004-A); 四川省科技支撑项目(2011RZ0003)

ANGA 按照适应值的大小 D 把种群 N 个个体分为 N/D 个子种群,在子种群中分别进行遗传算子操作,最终在集合所有个体选出适应值高的个体.这样保证了种群的多样性,也确保了种群的收敛性.

2.1 获取适应特征值

适应值反映种群的生存能力,在负载调度的应用中适应值表现每台服务器能接受的工作任务的能力.适应特征值是决定种群适应值关键因素,是适应值的自变量,所以通常在解决负载调度类问题中,由于服务器响应处理工作对服务器的中央处理单元(CPU)要求最高,所以适应值由 CPU 利用率为主要参数计算最为恰当.但是实时的 CPU 利用率只能反应服务器当前的负载情况,并不能准确的反应服务器在未来的某段时间能承受的负载的情况,所以本文提出相对 CPU 利用率作为适应特征值来计算适应值.所谓相对 CPU 即单位时间内的 CPU 的变化率.本文用特有的进程(monitor)接收并保存当前 CPU 利用率,并与上一次记录下的 CPU 利用率相减即得到相对 CPU 利用率,即为

$$\Delta \text{cpu} = \text{cpu}[k] - \text{cpu}[k-1]$$
 (1)

式中, k 为第 k 次记录下的 CPU 利用率.再根据以下公式^[5]:

$$\text{sub_Load}_{(i)} = (X_{(i)} + 1) \times \Delta \text{cpu}_{(k)}$$
 (2)

$$\text{aver_Load} = \sum \text{sub_Load}_{(i)} / N$$
 (3)

$$\text{sub_error}_{(i)} = \text{sub_Load}_{(i)} - \text{aver_Load}$$
 (4)

$$\text{Error} = \sqrt{(\sum \text{sub_error}_{(i)}^2) / \sum \text{sub_Load}_{(i)}}$$
 (5)

式中, i 表示第 i 台服务器的序列号.计算出适应值

$$\text{Fit_value} = 1 - \text{Error}$$
 (6)

2.2 算法框架

- 本文采用改进的小生境遗传算子步骤如下
- 第 1 步:随机生成规模为 N 的初始种群;
 - 第 2 步:通过“轮盘赌”进行个体选择;
 - 第 3 步:根据获取的 CPU 和内存占用率计算出初始结群的个体适应值,并排序;
 - 第 4 步:如果满足条件执行步骤 5,如果不满足条件则退出;
 - 第 5 步:计算大众群个体方差 D ,若 $D \geq \sigma$ 则设置子种群规模 $K = f(D)$,否则 $K = 2$;再选取相邻 K 个个体形成子种群; $(f(D)$ 为高斯数);
 - 第 6 步:子种群内个体进行均匀模板杂交,并进行 $(u + \lambda)$ 选择;
 - 第 7 步:进行随机变异,若变异个体为最优个体,则进行 $(1 + 1)$ 选择;

- 第 8 步:直转步骤 3
- 第 9 步:输出结果;

3 架构设计

定义 1 决策器是用于服务器集群中训练调度序列的服务器.它通过接收实体服务器的负荷状况等信息,产生进程训练调度序列.

由于 ANGA 进化代数较大,尤其是在系统节点较多的情况下导致 LB 的运算量极大,严重的影响其调度速度,本文在架构设计上增加了一个专门负责计算的节点,该节点在本文中被称为决策器.在上文中所提到的 monitor 进程即是在决策器上执行.决策器每隔一段时间会收集一次真实服务器的负荷状况的信息(CPU 占用率)作为遗传算法的输入,并在遗传算法运行结束后将结果序列发送给 LB.这种架构使得 LB 的动态调度的工作量由决策器共同分担.系统架构如图 1 所示.

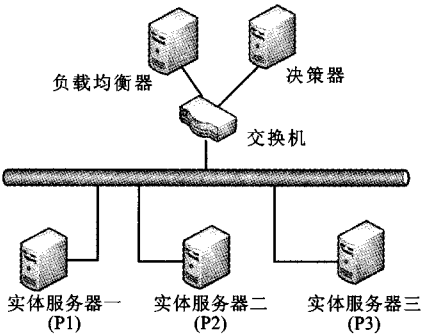


图 1 系统框架图

为了让 LB 根据决策器提供的训练后的序列来进行调度,本文在 Linux 系统的 ipvs 模块中增加了一个 drr(dynamic round robin)调度算法模块. drr 算法的输入即是神经网络算法训练出的序列,这个序列即是在某一段时间内对实体服务器的调度序列. drr 算法需要将输入序列转换为相对应的可识别的 ip 地址.改进后系统调度流程如图 2 所示.

4 实验测试

为了证明算法的有效性,本文进行了模拟实验. 本文将改进的动态负载算法加入到 Linux 内核的 ipvs^[6]模块中,并增加了一个新的调度算法.实验采用网络地址转换(NAT)^[7]模式布局,三台 pc 机作为实体服务器(P1、P2、P3),各服务器的配置如表 1 所示.

表 1 测试服务器配置主要参数表

LB	moniter	P1	P2	P3
Xeon 5500 8G	Core i3 2G	Core i3 1G	Core i5 2G	Core i3 2G

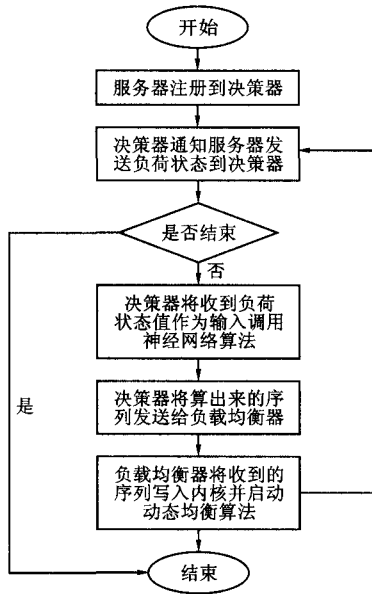


图2 系统调度流程图

本文采用的压力测试工具是 Jmeter, 同时 Wireshark 辅助产生混合流量来模拟网络环境产生的大量负载. 算法的主要参数设置有: 种群数 N 为 30, 最大进化代数 500; 交叉概率 P_c 为 0.8, 变异概率 P_m 为 0.05. 算法改进是针对用户访问服务器资源不同, 导致服务器负载倾斜的情况, 因此测试时需要让客户端访问不同的资源, 从而给服务器产生不同的压力. 在服务器端设置测试网站设置了三种不同类型, 一是普通的 PHP 页面, 它返回的速度最快, 二是对数据库有 50 000 次插入操作的 PHP 页面, 三是类似于第二类页面, 但是插入操作次数是第二类的 2 倍, 因此, 第三类页面访问脚本消耗的服务器资源最多, 给服务器产生的压力最大. 由于不能测出每台服务器实时的响应时间, 表 2 中列出了每台服务器在静态轮询算法(round robin, rr)、静态加权最小链接(weighted least connections, wlc)^[7]、drr 三种分配策略的平均响应时间和错误率. 在实验中记录下每台服务器 CPU 利用率情况, 图 3 为三台服务器在不同策略情况下拟合出的 CPU 利用率方差对比图.

表 2 三种算法系统响应对比表

	drr	rr	wlc
平均响应时间/ms	2	9	18
错误率/%	0	0.268	0.768

从表 2 看出, 算法及架构改进后使系统响应时间大大缩短, 同时错误率降为 0. 图 3 中统计了在三台服务器下三台实体服务器的 CPU 利用率的方差.

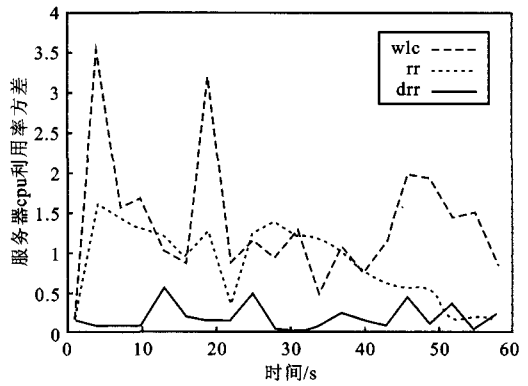


图3 服务器cpu利用率图

不难看出, drr 算法的方差远远小于其他两种算法, 这说明 drr 算法作为调度算法, 其系统调度的均衡性能远远高于 wlc 和 rr 算法. 尤其是在 5 s 和 20 s 这两个时间点, wlc 和 rr 算法的方差曲线明显上升到一个高峰点, 这是由于此时系统调度没有考虑服务器实际处理能力而调度. 而 drr 算法由于其自适应性能, 方差曲线较为平滑. 说明了 drr 算法在处理调度问题上, 能更好地使服务器均衡地利用. 这一优势尤其在实体服务器工作任务过于消耗 CPU 的情况下(如: 更新大型数据库等操作)更为突出. 其中主要原因, 一是改进的遗传算子能保持样本的多样性, 因此解是通过更大幅度的搜索而得到的, 所以训练出的序列更具更优于一般遗传算法; 二是系统模式有所改进, 将收集数据和训练数据集工作集中于一台服务器完成, 这样减小了算法带来给 LB 的 CPU 利用率的消耗. 通过模拟实验分析表明, 本文的改进算法和改进的系统调度模式, 能在集群环境中以较短时间实现 Web 请求的快速分配.

5 结束语

本文采用了 ANGA, 并结合负载均衡调度的具体问题, 提出了对服务器集群架构和传统的遗传算子改进, 使 ANGA 恰当的应用于服务器负载均衡结构的. 通过实验证明, 改进的负载均衡系统能弥补传统的遗传算法的缺点, 很好地完成系统的负载均衡调度.

参考文献:

[1] 余燕芳, 陆军. 基于改进遗传算法的服务器端负载均衡算法[J]. 微电子学与计算机, 2007, 24(7): 146-148.
[2] 段淮川, 胡平. 基于剩余负载率的动态负载均衡研究[J]. 微电子与计算机, 2010, 27(2): 141-144.

(下转第 60 页)

对 DSP 的调试. 调试过程的关键在于对故障点的定位^[5]. 本文采用了模拟器作为 Gold Model 与硬件进行比较, 调试过程如图 5 所示. SuperV 调试平台对触发事件进行检测, 将此事件状态下的执行信息反馈给串口 API 中的比较模块, 与此同时, 模拟器亦将相应的信息发送给比较模块, 比较模块通过比对来判定两者信息是否一致, 如果一致, 则说明 SuperV 运行准确无误, 如果不一致, 则将故障信息通过串口反馈给 PC.

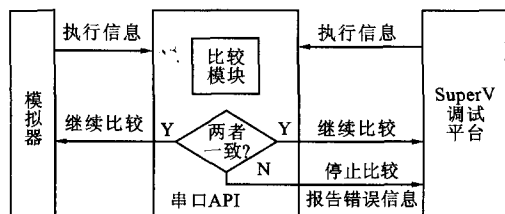


图 5 调试过程

在比较过程中, 可以将执行信息的比较划分为细粒度的指令级、中粒度的循环级与粗粒度的函数级. 在比较初期, 本文首先对函数级进行比较, 以一种粗粒度的方式来确定故障点的大致范围; 然后在此范围内依次对循环级与指令级进行中粒度与细粒度比较, 最终对故障点进行定位.

使用本平台, 在 mp3 解码程序、mpeg2 解码程序、语音识别程序等应用程序出现故障时, 本平台实现了对故障点的快速定位, 最终完成了对这些应用程序的调试, 使其能正确地运行在 SuperV 处理器上.

6 结束语

本文介绍了一种基于串口的 DSP 调试平台设

计方法. 通过以串口通信基础的驻留程序, 实现了对应用程序的加载, 以及上位机对应用程序执行的灵活控制. 在驻留程序中增加了校验机制, 确保了所传输数据的正确性. 本文设计的调试平台具有良好的反馈特性, 通过上、下位机的信息实时交互, 实现了协同调试.

参考文献:

- [1] 吴琼之, 南方, 张峰. 并行 DSP 系统软件调试器设计与实现[J]. 北京理工大学学报, 2011, 31(7): 855-858.
- [2] 高瑛珂, 王琪, 李泉泉, 等. 一种基于 JTAG 接口的片上调试与性能分析方法[J]. 微电子学与计算机, 2012, 29(7): 68-71.
- [3] 罗云虎. 一种利用 Win32 API 函数实现串口实时通讯的方法[J]. 现代电子技术, 2004, 27(11): 70-73.
- [4] 王彦臣, 凌伟, 黄廉卿. 基于 Client/Server 模式下实现大数据量文件的传输[J]. 光学精密工程, 2003, 11(4): 400-403.
- [5] Wang Qi, Zhu Hao, Gao Yingke, et al. Accelerating processor verification based on ESL model[C]// Recent Advances in Computer Science and Information Engineering. Berlin: Springer-Verlag, 2012: 745-751.

作者简介:



薛志远 男, (1986—), 博士研究生. 研究方向为集成电路验证.

(上接第 56 页)

- [3] 孙林, 罗大庸. 基于遗传算法的 Web 服务器集群负载均衡研究[J]. 计算机测量与控制, 2006, 14(10): 1364-1365.
- [4] 陕宣耀, 王芳. 一种改进的小生境遗传算法[J]. 重庆邮电学院学报, 2005, 17(6): 721-723.
- [5] 黄聪明, 陈湘秀. 小生境遗传算法的改进[J]. 北京理工大学学报, 2004, 24(8): 675-678.

- [6] 王强. 基于 LVS 集群的一种负载均衡改进算法的研究与实现[D]. 成都: 电子科技大学, 2006.
- [7] 章文嵩. 可伸缩网络服务的研究与实现[D]. 北京: 国防科技大学, 2000: 109.

作者简介:

杨越 女, (1987—), 硕士研究生. 研究方向为计算机网络通信.