文章编号：1007-5321(2017)增-0093-05 DOI:10.13190/j.jbupt.2017.s.021

结合分类回归树和 K 近邻的负载均衡预测算法

朱 斌， 孙 斌

（北京邮电大学 信息安全中心，北京 100876）

摘要：提出了针对移动平台使用 XMPP 协议服务器端的基于分类回归树和 K 近邻结合的预测算法. 该方法首先通过动态反馈采集服务器节点的资源信息组成时间序列，对时间序列进行预测计算. 然后将服务器节点分区域管理，运用不同的调度策略. 实验结果证明，与原始的加权轮询和最小连接数算法相比，该预测算法在连接响应时间上减少了 25%，在建立连接的平均速率上提升了近 1.3 倍，动态的调度策略使得服务器集群有更大的吞吐量，对于移动平台有更好的适应性.

关 键 词：负载均衡；时间序列的预测；分类回归树；K 近邻；动态调度策略

中图分类号：TP393.1 **文献标志码**：A

A Load Balancing Predication Algorithm of CART and KNN

ZHU Bin， SUN Bin

(Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: To address the problems in the mobile platform based on XMPP protocol, a prediction method of the server load based on classification and regression tree and K-nearest neighbor machine learning algorithm was presented. The algorithm made up time series of load by gathering every node's load information comprehensively and analyzed the time series, to carry out prediction. And then, the server nodes were divided into three regions, different scheduling strategies were used in different regions. Simulation experiments and tests showed that compared with the weight round robin and least connection algorithm, this proposed prediction algorithm decreased connection response time by 25%, and increased the connection establishment by 1.3 times. Dynamic scheduling strategy made the communication server cluster has a greater network throughput, which has more robust adaptability for mobile platform.

Key words: load balancing; forecast of time series; classification and regression tree; K-nearest neighbor; dynamic scheduling strategy

由于蜂窝网络和 WiFi 网络的状态会出现异常改变，导致移动平台资源出现频繁的离线、掉线等情况，需要服务器能够快速处理客户端请求，又要求其在较短的时间内做出有效应答响应. 结合以上分析，提出的负载均衡预测算法及调度策略实现流程如下.

1）利用资源采集程序全面地收集服务器节点的负载，得出每个结点的负载时间序列值.

2）使用基于回归树与 K 近邻[1-3]方法来学习预测模型，提出一种适用于通信集群节点的负载时间序列的平滑处理方法，得出精度更高的负载预测值.

收稿日期：2016-05-29

基金项目：国家 242 信息安全计划项目(2015A136)

作者简介：朱 斌(1992—)，男，硕士生，E-mail：bupt_zhubin@163.com；孙 斌(1967—)，女，副教授.

3）对服务器节点进行区域划分，不同区域采用不同的动态调度策略.

1 动态负载均衡算法的实现

1.1 基于回归树和 KNN 结合的预测算法实现

采集静态和动态资源信息，周期 T 为 10 s. 如表 1 和表 2 所示.

表 1 静态参数名称表

静态资源	参数表示
地理位置	S_{place}
ISP	S_{isp}
网络接入类型	S_{type}
CPU 核数	S_{cpu}
内存大小	S_{memory}
带宽大小	$S_{\text{bandwidth}}$
磁盘大小	S_{disk}

表 2 静态参数名称表

动态资源	参数表示
CPU 使用率	L_{cpu}
内存使用率	L_{memory}
接口流入包数	L_{in}
接口流出包数	L_{out}
进程数	L_{process}
TCP 连接数	L_{tcp}
打开的端口数	L_{port}

假设服务器的综合性能指标为 $S(C_i)$，C_i 表示第 i 个服务器节点，$i \in (1,2,\cdots,N)$.

$$S(C_i) = k_1 S_{\text{place}}(C_i) + k_2 S_{\text{isp}}(C_i) + k_3 S_{\text{type}}(C_i) + k_4 S_{\text{cpu}}(C_i) + k_5 S_{\text{memory}}(C_i) + k_6 S_{\text{bandwidth}}(C_i) + k_7 S_{\text{disk}}(C_i)$$

$\sum_{i=1}^{7} k_i = 1$，k_i 表示各个指标的权重系数.

假设服务器节点的动态负载值为 $L(t_i)$，t_i 表示第 i 个服务器节点，$i \in (1,2,\cdots,N)$.

$$L(C_i) = t_1 L_{\text{cpu}}(C_i) + t_2 L_{\text{memory}}(C_i) + t_3 L_{\text{in}}(C_i) + k_4 L_{\text{out}}(C_i) + k_5 L_{\text{process}}(C_i) + k_6 L_{\text{tcp}}(C_i) + k_7 L_{\text{port}}(C_i)$$

$\sum_{i=1}^{7} t_i = 1$，t_i 表示各个参数的权重值.

结合服务器综合性能 S 和动态负载值 L 这两个重要参数来精确地评估当前服务器节点的负载能力. 假设权值为 X，X 关于服务器性能和动态负载值两个参数的计算方法如下：

$$X(C_i) = \frac{L(C_i)}{S(C_i)}$$

通过计算得出的负载的时间序列 $\{X(1), X(2), X(3), \cdots, X(N)\}$ 负载时间序列预测[4]是根据时间序列的历史数据 $\{X(t-1), X(t-2), X(t-3), \cdots, X(t-m+1)\}$ 预测未来 $t+\tau$ 时刻的值 $X(t+\tau)$. 参数 τ 是时间延迟.

分析文献[5-6]的平滑算法设计，提出一种部分的分支平滑算法. 定义 n_i^j 为节点上的所属类别 i 的基数值. 定义 $p_i^0 = \frac{1}{c}$. 通过下面的公式，就可以计算从 V_1 到 V_d 的所有可能性.

$$p_i^j = \frac{n_t^j + m p_t^{j-1}}{\sum_{i \in c} n_i^j + m}$$

其中：p_i^d 为最底层叶节点的可能性，m 为平滑参数. m 设定为 $K, 2\times K, 3\times K$ 中的最优值. 设定一个阈值 θ，即当前节点的权重低于预先设定的阈值 θ 时，就不再进一步向上平滑. 因为随着从叶节点返回根节点的路径上，每次返回父节点时其训练样本数越来越大，相应地对预测样本的目标值的影响权重却越来越小. 当权重值小于预设阈值时，就认为当前节点以及通往根节点的路径上的所有节点对最终的预测目标值的影响可以忽略不计. 充分地利用从根节点到相应结果类的叶节点的路径上的节点信息，使得预测的结果更准确，并且使得预测在可能的某一个节点上停止，减少时间复杂度，使预测更精准更快速.

1.2 移动平台的负载均衡调度策略

根据得出的服务器节点负载预测结果，将所有的服务器节点按照负载预测值分为 3 个区域：空闲区、正常负载区、过载区. 每个服务器节点被认为具有 3 种状态：空闲、正常、过载. 根据其负载情况，将其分配到对应的区间中. 将所有预测得到的服务器节点的负载值进行求和平均.

$$\text{Load\_Degree}_{\text{avg}} = \sum_{i=1}^{n} \frac{P_i}{n}$$

空闲区的服务器节点状态：Load_Degree < Load_Degree$_{avg}$ × 50%.

正常负载区的服务器节点状态：Load_Degree$_{avg}$ × 50% < Load_Degree < Load_Degree$_{avg}$ × (1 + 50%).

过载区的服务器节点状态：Load_Degree > Load_Degree$_{avg}$ × (1 + 50%).

在负载均衡器上会维护一张负载状态表，在固定的每个周期 T 内更新一次，该表管理了 3 个区域内的所有服务器节点. 每当一个新的请求到达时，会根据这张负载状态表，选定一个区域，使用该区域的调度策略选取一台服务器节点进行分配服务.

1）空闲区域内的调度策略

当来自移动平台的请求连接到来的频率很快，在短时间内会集中到来时，将在空闲区域内通过一个快速的调度算法，能够保证可以以很快的速度分配新来的连接请求.

提出一个基于优先级队列的轮询调度算法. 优先级队列每次从队列中取出的是具有最高优先级的元素. 调度过程如下.

① 在轮询调度的前一步，即创建队列时，将空闲区域内的节点按负载值递增排序，将负载值较小的标记为较高的优先级. 即负载值最小的节点拥有最高优先级，负载值最大的节点拥有最低优先级.

② 生成一个循环队列并且在分配连接时循环遍历该队列.

③ 每次从优先级队列中，按照优先级情况取出对应的服务器节点来提供服务. 节点的负载情况会在每个周期 T 内更新一次，节点优先级的情况也会实时变化.

2）正常负载区域内的调度策略

当请求连接到来的频率相对较慢，要保证在正常负载区域内得到一个负载值递增有序的节点序列，并且对于该区域的节点添加和删除要有较小的时间复杂度完成，因此考虑引入基于改进的红黑树排序的调度策略. 对红黑树进行改进，使得该特殊的二叉排序树支持值相等的节点存在. 由于红黑树的特有性质，可以使得整棵二叉树接近平衡二叉树，使得最终的查找、插入和删除的时间复杂度为 $O(\log n)$. 负载均衡的调度步骤如下.

① 对正常区域内的服务器节点，根据服务器节点的负载预测值进行遍历，将此服务器节点入树，构建一棵改进的红黑树.

② 得到该红黑树后，对红黑树进行中序遍历，形成一个递增的数列，服务器节点的负载预测值将形成一组由小到大的递增数列.

③ 当来自移动平台的请求到来时，选择具有较小负载的且满足条件的服务器进行连接分配. 服务器节点分配好请求连接后，它的负载值也会发生变化，下一周期会重新调整得到一个新的红黑树.

④ 在周期 T 内，出现有服务器的负载超过正常负载的区间，则将该服务器节点从红黑树上删除，将删除的节点移入过载区间.

⑤ 在周期 T 内，如果过载区域里的服务器节点的负载值下降到正常状态，则将该服务器节点从过载区域中删除，动态得添加新的服务器到正常可用区域内.

2　实验方案与结果分析

针对负载均衡服务器分别采用 3 种负载均衡算法：加权轮询调度，最小连接数算法，和提出的基于分类回归树（CART，classification and regression tree）和 K 近邻（KNN，k-nearest neighbor）算法结合的动态负载均衡算法，使用 Tsung 分别模拟 1 000、5 000、10 000 用户的注册和登录，向负载均衡调度服务器发起 XMPP 请求连接. 从多个角度来分析得出该算法在连接响应时间、连接建立速率、吞吐量等方面的优越性. 在用户连接登录数为 1 000、5 000、10 000 的情况下，每 0.5 s 模拟一个用户注册并登录，算法性能的对比，如表 3 ~ 表 5 所示.

表 3　连接登录数为 1 000 的性能对比

负载均衡算法	连接响应时间/ms	建立连接平均速率/s^{-1}
加权轮询	86.03	1.19
最小连接数	65.80	1.52
CART&KNN	60.12	1.83

表 4　连接登录数为 5 000 的性能对比

负载均衡算法	连接响应时间/ ms	建立连接平均速率/s^{-1}
加权轮询	410.00	9.67
最小连接数	315.38	12.39
CART&KNN	242.60	15.85

表5 连接登录数为10 000的性能对比

负载均衡算法	连接响应时间/ ms	建立连接平均速率/s^{-1}
加权轮询	2 350.0	21.10
最小连接数	1 865.7	27.20
CART&KNN	1 382.0	38.08

采用了本预测算法的调度策略,预先获得各节点的负载变化,减少了由于负载信息传递时延带来的误差,使得响应的时间能够有效的缩减. 从连接响应时间的角度,该算法较原始的算法减少了近25%的响应时间.

从建立连接的平均速率可以看到,本预测算法在登录数为1 000、5 000、10 000的模拟过程中,速率都在不断提升,在短时1万个用户访问到来,在每秒能够建立38个请求连接,相对于原始的最小连接数算法有近1.3倍的速率提升,有更快的处理速度,更强的适应能力. 在压力测试时间变化的情况下,对整个通信服务器集群的吞吐量进行了采集获取,并使用gnuplot对吞吐量的变化进行了绘图分析,如图1~图3所示.

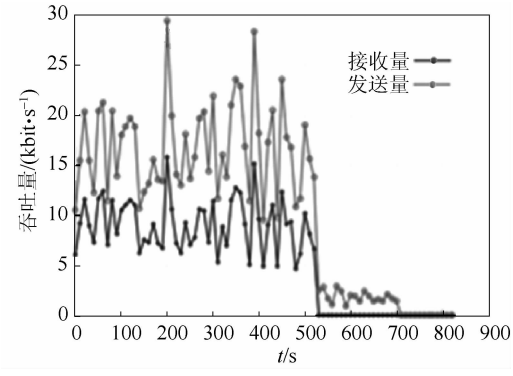

图1 压力量为1 000时的吞吐量

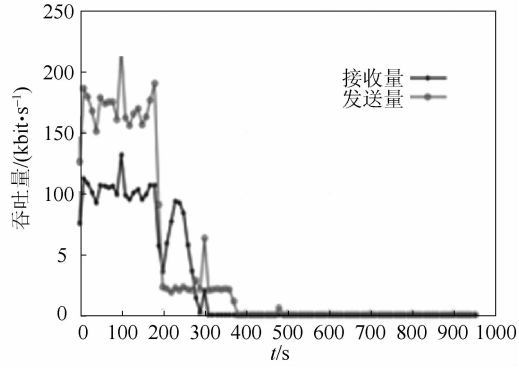

图2 压力量为5 000时的吞吐量

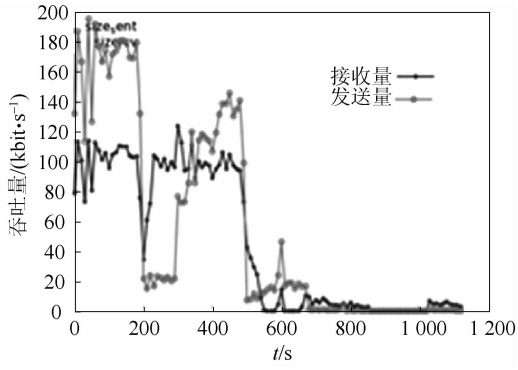

图3 压力量为1 000的用户情况的吞吐量

上面的3幅图中的横坐标表示该压力测试的时间,纵坐标的单位是bit/s,接收量的带圆点曲线是响应的总大小,发送量的带断点曲线是发出请求的总大小. 对于请求连接的增加,从1 000~5 000再到1万的连接量,系统表现的吞吐量也随着增加,并且能较长时间维持较大的吞吐量. 系统的吞吐量在高水平维持的时间总体较长,平均吞吐量在140 kbits/s,体现了本预测算法和动态调度策略在对于系统的资源利用率上能够充分使用系统集群的空闲能力,将节点的资源能够利用起来,比原始的多种静态和动态算法能够有更大的吞吐量,适应于环境更复杂,连接更频繁的移动平台情况.

3 结束语

经过实验分析,该算法对于负载值的预测更精准,动态调度策略的适应能力更强. 对于原始的加权轮询和最小连接数算法,在连接响应时间上减少了25%,在连接建立的平均速率上提升了近1.3倍的处理能力,有着更强的适应性和吞吐能力.

参考文献:

[1] 郭昌辉, 刘贵全, 张磊. 基于回归树与K-最近邻交互模型的存储设备性能预测[J]. 南京大学学报: 自然科学版, 2012, 48(2): 8-17.
Guo Changhui, Liu Guiquan, Zhang Lei. An interactive model based on regression tree and K-nearest neighbor for storage device performance prediction[J]. Journal of Nanjing University: Natural Sciences, 2012, 48(2): 8-17.

[2] Pinem A F A, Setiawan E B. Implementation of classification and regression Tree (CART) and fuzzy logic algorithm for intrusion detection system[C]// Information and Communication Technology (ICoICT), 2015 3rd In-

ternational Conference on. [S. l.]: IEEE, 2015: 266-271.

[3] Yigit H. A weighting approach for KNN classifier[C]// 2013 International Conference on (ICECCO) Electronics, Computer and Computation. [S. l.]: IEEE, 2013: 228-231.

[4] Dinda P A. The statistical properties of host load[J]. Scientific Programming, 1999, 7(3-4): 211-229.

[5] Weiss S M, Indurkhya N. Rule-based regression[C]// IJCAI. 1993: 1072-1078.

[6] Ren Xiaona, Lin Rongheng, Zou Hua. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast[C]// 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS). Beijing:IEEE,2011.

(上接第 84 页)

参考文献：

[1] 董蓉，李勃，陈启美. 路况视频中 HSV 彩色不变量阴影检测法研究与改进[J]. 中国图象图形学报，2009，14(12)：2483-2488.
Dong Rong, Li Bo, Chen Qimei. Research and improvement on shadow detection in expressway videos using HSV color model[J]. Journal of Image and Graphics, 2009, 14(12): 2483-2488.

[2] 高俊祥，杜海清，刘勇. 采用光照不变特征的椭球法运动阴影检测[J]. 北京邮电大学学报，2009，32(5)：109-113.
Gao Junxiang, Du Haiqing, Liu Yong. Moving shadow detection by ellipsoidal method using illumination invariants[J]. Journal of Beijing University of Posts and Telecommunications, 2009, 32(5): 109-113.

[3] 解文华，易本顺，肖进胜，等. 基于颜色和区域梯度方向特征的阴影检测算法[J]. 中南大学学报(自然科学版)，2013(12)：4874-4880.
Xie Wenhua, Yi Benshun, Xiao Jinsheng, etc. Shadow detection algorithm based on color and regional gradient direction features[J]. Journal of Central South University (Science and Technology), 2013(12): 4874-4880.

[4] 邱一川，张亚英，刘春梅. 多特征融合的车辆阴影消除[J]. 中国图象图形学报，2015，20(3)：0311-0319.
Qiu Yichun, Zhang Yaying, Liu Chunmei. Vehicle shadow removal with multi-feature fusion[J]. Journal of Image and Graphics, 2015, 20(3): 0311-0319.

[5] Satpathy A, Jiang X, Eng H L. LBP-based edge-texture features for object recognition[J]. IEEE Transactions on Image Processing a Publication of the IEEE Signal Processing Society, 2014, 23(5): 1953-1964.

[6] Al-Najdawi N, Bez H E, Singhai J, et al. A survey of cast shadow detection algorithms[J]. Pattern Recognition Letters, 2012, 33(6): 752-764.

[7] 姜柯，李艾华，苏延召. 结合边缘纹理和抽样推断的自适应阴影检测算法[J]. 西安交通大学学报，2013，47(2)：39-46.
Jiang Ke, Li Aihua, Su Yanzhao. An adaptive shadow detection algorithm using edge texture and sampling deduction[J]. Journal of Xi'an Jiaotong University, 2013, 47(2): 39-46.