一种大规模分布式计算负载均衡策略

杨际祥[1,2],谭国真[2],王 凡[2],周美娜[2]
(1.重庆交通大学理学院,重庆 400074; 2.大连理工大学计算机科学与技术学院,辽宁大连 116024)

摘 要: 大规模和超大规模计算系统中的通讯延迟成为影响负载均衡性能的一个重要因素,且延迟具有时变性,而传统的负载均衡策略通常假设通讯固定不变或不考虑通讯延迟开销.本文考虑了系统的通讯延迟开销和延迟时变性特征,给出一种基于广义神经网络(GNN)的层次结构负载均衡策略.该策略具有三个特点:(1)通讯优化的层次结构能够降低大规模计算系统的负载均衡开销;(2)考虑了节点计算速率及通讯延迟时变特性;(3)通过延迟预测可优化任务的通讯延迟和迁移延迟开销.仿真实验验证了该策略在通讯和负载均衡开销方面的性能.

关键词: 大规模计算系统; 层次结构负载均衡; 广义神经网络 (GNN)
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2012) 11-2226-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.11.014

A Load Balancing Strategy for Large-Scale Distributed Computing

YANG Ji-xiang[1,2], TAN Guo-zhen[2], WANG Fan[2], ZHOU Mei-na[2]
(1. *School of Science, Chongqing Jiaotong University, Chongqing* 400074, *China*;
2. *School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning* 116024, *China*)

Abstract: Traditional load balancing strategies generally assume that communication delay is deterministic or the communication overhead can be ignored. A hierarchical load balancing strategy based on generalized neural network (GNN), considering the communication overhead and time-varying delay feature, is presented for large-scale computing systems. The strategy possesses three features: 1) load balancing overhead can be reduced with optimizing hierarchical structure communication; 2) considering the heterogeneity in the processing rates of the nodes and the delay randomness imposed by the communication medium; 3) optimizing task communication delay and migration delay. Simulation results demonstrate the capabilities.

Key words: large-scale system; hierarchical load balancing; generalized neural network

1 引言

在传统负载均衡策略中,由于计算过程中传输的任务粒度和跳数相对较小,一般都假定通讯为固定的,或者不考虑通讯开销[1~4].在大规模分布式计算环境中,网络直径、传输的粒度和传输的数据规模较大.例如,在云计算环境中,网络环境为 Internet 互联网,网络拓扑直径很大,调度目标通常为虚拟机(VM)资源,需要传输的任务粒度和数据规模都较大[5].大规模信息量的传输将导致较大的通讯延迟,无疑会降低调度信息的精确性和提高信息的陈旧性.

集中式负载均衡策略通过主节点或中心节点来收集全局的负载信息用于决策,大量的信息在中心节点积聚会产生较高的开销与延迟,使得主节点的通讯与处理速度很快慢下来.当这种延迟增大至某一阈值时,可导致信息失效,从而可导致错误的负载均衡决策.目前的商用产品基于集中式的策略[6~8],这些策略可扩展性低,不适合大规模的分布式计算环境,而分布式负载均衡策略可实现较高的可扩展性[9].然而,分布式的负载均衡策略仅仅通过使用局部的负载信息来进行负载均衡的决策,局部信息精确性不高,使得均衡器难以做出一个最优的负载均衡决策,尤其是在像 Internet 这样的快速变化的环境中,均衡器甚至会做出错误的决策,导致某些计算任务或调度对象在多个节点或处理器间来回迁移而迟迟得不到处理或计算,增加了额外开销,同时也可降低系统的响应速度和吞吐量等.这种延迟大大降低了动态负载均衡(DLB)策略的有效性,增大了响应时间.对于云服务的用户体验而言,体验质量好坏在很

收稿日期:2011-05-30;修回日期:2012-05-31
基金项目:国家 863 高技术研究发展计划(No.2012AA111902);国家科技支撑计划(No.2011BAK02B02-01);中央高校基本科研业务费专项资金(No.DUT11SX09);河北省科技支撑计划(No.11213575)

大程度上取决于用户访问的服务实例和终端用户之间的响应时间[10,11].这种响应时间主要由用户和托管实例的数据中心之间的互联网(Internet)延迟引起[11].

当系统中的节点存在大的延迟时,大规模的分布式处理和计算的综合性能降低[12].当通讯延迟较高时,且通讯时间成压倒性地高于计算时间之势时,极有可能使得不同域中的虚拟机中的性能不能获得任何的加速比[13].在云计算环境中,当前的虚拟机器资源调度主要考虑了系统的状态,而很少考虑系统的变化和历史数据,它总是导致系统的负载失衡[5].对于高性能计算,CPU 速度和通讯延迟是影响其性能的两个主要因素[14].

最近几年的研究发现,通讯延迟具有时变性[15],这是由于网络中的阻塞情况、通信量和一些未知因素所决定的[16,17].如何在负载均衡过程中快速而准确地感知或预测通讯状态并据此做出适时的决策成为大规模分布式计算系统需要解决的一个关键问题.目前,也存在一些求解该问题的方法,例如 RW 法、HA 法和 IHA 方法.其中,RW 仅仅基于当前的网络延迟状况来进行判断,HA 仅仅根据历史数据的平均状况来判断,IHA 则综合了前面两种方法.它们并不能反映延迟的非线性和不确定性特点,不能克服随机干扰因素产生的影响.因此,这些方法的预测精度随预测时间间隔的缩短而降低.

Dhakal 等[16]研究指出,在实际的大规模分布式计算系统中的负载均衡策略未考虑延迟随机性时的性能非常低下,并在文献[17]中给出了一个随机延迟预测公式.不过,该方法是 RW 法和 HA 法相结合的一种方法,其遗忘因子 α 值只能根据经验来选择,对于网络的动态变化情况是很难预测的,不具有实际可用性.此外,它不具有一个完整的预测模型,只能预测下一时间段的平均延迟,而不能预测下一时段之后的延迟.

根据大规模分布式计算系统中通讯延迟开销的时变性特征,本文给出了一种基于 GNN 的层次结构动态负载均衡(GNNDLB)策略.该策略通过通讯优化的层次结构来降低大规模分布式计算系统的负载均衡开销;考虑了节点的计算速率和通讯延迟时变特性,基于 GNN 理论构建延迟预测模型,优化延迟开销,为考虑延迟的负载均衡策略提供一种有效的优化方法.

2　一种层次结构负载均衡策略

2.1　智能神经元模型

本文引入一种基于线性独立函数的智能神经元模型,它较普通神经元具有更高的知识存储能力,可使整个神经网络的信息处理能力得到增强.将由这种智能神经元组成的 GNN 应用于大规模并行与分布式计算系统的延迟量预测,具有更高的实用性.

在以前的研究中,人们用线性独立函数对神经网络输入层进行预处理以使神经网络具有较好的映射效果[18],这些函数在不引入新的信息情况下,能够有效地增加变量的空间维数,从而提高网络收敛速度.本文将线性独立函数引入到智能神经元内部构造中,将神经元输入 x 扩展为线性独立的函数 $x, x^2, x^3, \cdots, x^n$,构建新的神经元模型.其中,x 为神经元输入,$\boldsymbol{a}_r$、$\boldsymbol{b}_N$ 为可调变量,$\boldsymbol{F}(x)$为神经元输出.$\boldsymbol{F}(x)$可用式(1)来表示:

$$\boldsymbol{F}(x) = \sum_{r=1}^{n} \boldsymbol{a}_r x^r + \boldsymbol{b}_N \tag{1}$$

设神经元具有 n 个输入样本,有如式(2)所示的映射关系:

$$\boldsymbol{XA} + \boldsymbol{b}_N = \boldsymbol{Y} \tag{2}$$

在上式中,$\boldsymbol{Y}$ 为神经元输出,$\boldsymbol{X}$ 可表示为下式:

$$\boldsymbol{X} = \begin{bmatrix} x_1 & x_1^2 & \cdots & x_1^n \\ x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}, \boldsymbol{A}^{\mathrm{T}} = (a_1, a_2, \cdots, a_n)$$

对上式中的 $\boldsymbol{X}$ 提取公因子,可得式(3):

$$\boldsymbol{X} = \prod_{r=1}^{n} x_r \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix} \tag{3}$$

由范德蒙(Vandermonde)行列式可得式(4):

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix} = \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i) \tag{4}$$

因此,矩阵 $\boldsymbol{X}$ 的值可表示为式(5):

$$\det \boldsymbol{X} = |\boldsymbol{X}| = \prod_{r=1}^{n} x_r \prod_{1 \leqslant i < j \leqslant n} (x_j - x_i) \tag{5}$$

因为 $x_i \neq x_j, \det \boldsymbol{X} \neq 0$,矩阵 $\boldsymbol{X}$ 的秩为 n,所以 $\boldsymbol{A}^{\mathrm{T}} = (a_1, a_2, \cdots, a_n)$的解存在.说明神经元内部在不增加变量的情况下通过扩展函数的方法,增加了输入变量的维数,增强了神经元知识存储能力,能得到更好的函数映射效果.

有关学者应用正余弦函数组(x, $\sin\pi x$, $\cos\pi x$, $\sin 2\pi x$, $\cos 2\pi x$, $\cdots$)扩展神经网络输入[19],使神经网络映射效果得到增强.本文将正余弦函数组引入到智能神经元的构建中,建立如式(6)所示的智能神经元模型.

$$\boldsymbol{F}(x) = \sum_{r=1}^{k} \boldsymbol{a}_r S(r, x) + \boldsymbol{b}_N \tag{6}$$

由 $x, \sin\pi x, \cos\pi x, \sin 2\pi x, \cos 2\pi x, \cdots$这些线性独立函数组成的神经元的函数映射能力较好,相对于使用抽样函数作为基础函数的神经元[20]以及通过 $x, x^2,$

…,x^n 这些函数构成的神经元,其在收敛速度与预测精度方面较优.

2.2 基于 GNN 的延迟预测模型

通常说来,时间预测可以分为数据模型和分析模型.后者以数据指导为基本特点,通常以历史的和现在的延迟时间变量序列作为输入.假设当前时间为 t,则在时刻$(t-1),(t-2),\cdots,(t-n)$的历史时间数据分别为$f(t-1),f(t-2),\cdots,f(t-n)$.通过分析历史数据样本,能够预测未来的时间序列$f(t+1),f(t+2),\cdots$.

本文采用基于智能神经元模型的 GNN 方法实现节点在未来时段内的延迟预测.GNN 有多种结构形式,本文应用普通神经元组成 GNN 的输入层,应用智能神经元组成其隐含层与输出层.在对节点延迟量进行实时预测时,节点上的延迟与该节点前几个时刻的延迟量有着必然的联系,这样就可以利用该节点前几个时刻的延迟量数据预测该节点未来时刻的延迟时间.本文选取预测节点前六个连续时刻 $t-6\sim t-1$ 的延迟量作为 GNN 模型的输入,输出为预测节点 $t+1$ 时刻的延迟量,选取隐含层节点数为 6,进而建立基于 GNN 的延迟预测模型.

层与层之间的连接权值通过误差反向传播算法来学习,隐含层和输出层智能神经元内部可调参数的学习采用 LMS 算法.

2.3 基于 GNN 的层次结构负载均衡策略

通过 Yagoubi 的树型模型[21]将大规模分布式计算系统中的节点映射为一个层次树,并假设在时刻 t 从节点 i 到节点 j 的空载传输延迟为$\lambda_{ij,t}(0)$,则有式(7):

$$\lambda_{ij,t}(0)\equiv ps_{ij} \tag{7}$$

在式(7)中,$t=0$,p 表示相邻节点之间的空载延迟常量,s_{ij}表示从节点 i 到节点 j 的最短路径跳数.所以,在具有 N 个节点的分布式系统中,通过对每一层节点按式(8)

$$\lambda_{ij,t}(0)\leqslant\varphi \tag{8}$$

设定空载延迟阈值,确定负载均衡树的高度,构成一个负载均衡树.在该负载均衡树中,每一层的节点与其子节点构成一个负载均衡域,使用集中式的负载均衡策略,每一层的负载均衡策略通过阈值设定来压低负载均衡树的高度.具体算法(GNNDLB)描述如算法 1 所示.

2.4 负载均衡过程

(1)发起负载均衡.在一个设定的同步时刻,收集各叶子节点的信息,在负载均衡树中的每一个叶子节点将其上的任务的预计完成时间 T_i 发送给管理节点.如果一个管理节点具有 n 个叶子节点,那么这些子节点的状态可通过一个状态向量 $\boldsymbol{I}=(i_1,i_2,\cdots,i_n)$来表示,$i_i((0,1)$,其初始状态为$(0,0,\cdots,0)$.如果管理节点接收到节点 i 发送上来的 T_i,则在状态向量 $\boldsymbol{I}$ 中将 i_i 的值从 0 改为 1.

(2)判定负载失衡.当状态向量 $\boldsymbol{I}$ 的状态为(1,1,…,1)时,计算各个负载的期望完成时间,当满足式(9)时发起负载均衡.

$$\sigma(T)=\sqrt{|E(T^2)-[E(T)]^2|}\geqslant\eta \tag{9}$$

(3)计算每个叶子节点的负载超载量.任一节点的负载超载量可表示为式(10):

$$L_j^{ex}(t)=Q_j(t)-\frac{\lambda_j^d}{\sum_{k=1}^{n}\lambda_k^d}\sum_{l}^{n}Q_l(t) \tag{10}$$

在式(10)中,$Q_j(t)$表示在时刻 t 节点 j 上的 CPU 队列长度,λ_j^d 表示节点 j 的计算速率.如果 $L_j^{ex}(t)>0$,则该节点为超载节点;否则,该节点为低载节点.

(4)启动 GreedyCommLB 策略完成任务迁移.

算法 1 GNNDLB 算法

```
begin
  Data: V_t (任务集), E_o (通信图);
        V_p (处理器集), G_p (背景负载);
  Result: M: V_t → V_p;
  nObjs←NumObjs;
  ObjHeap objHeap(nObjs + 1);
  V_t→maxhAllObj;
  ProcessorHeap lightProcessors(P);
  G_p→lightProcessors;
  for i←1 to nObjs do
    minLoad←MAX_DOUBLE;
    o←objHeap.deleteMax();
    cpuDonor←lightProcessors.deleteMin();
    comm_cost←
    ∑_{e_ob∈E_o∧M(b)=cpuDonor} GNN(c_ob^);
    new_load←cpuDonor.load+comm_cost;
    if new_load<minLoad then
      minLoad←new_load;
      newDonor←cpuDonor;
    for cpuDonor∈P_comm do
      comm_cost←
      ∑_{e_ob∈E_o∧M(b)=cpuDonor} GNN(c_ob^);
      new_load←cpuDonor.load+comm_cost;
      if new_load<minLoad then
        minLoad←new_load;
        newDonor←cpuDonor;
    o→newDonor;
    调整更新 objHeap, lightProcessors;
  if objHeap.deleteMax()>sit_max then
    均衡失败, 发起更高层次的负载均衡;
  else
    负载均衡成功;
end
```

3 实验

3.1 GNNDLB 与传统集中式策略性能比较

本文通过 CHARM++ 的 BigSim[22]仿真器在深腾 1800 集群上使用 8 个节点来模拟 BlueGene/L(内存为

512MB,拥有 32K 到 64K 个处理器),应用负载均衡基准测试程序(lb_test)来模拟并行程序的运行.该程序在一个二维网孔结构的系统中生成特定数量的通讯对象或任务[23],每个对象完成一定数目的迭代.表 1 和表 2 分别给出了在模拟的 32K 和 64K 个节点上应用集中式结构和层次结构的 GreedyCommLB 策略完成一步迭代的负载均衡开销和内存开销.

表 1　32K 个模拟处理器上的策略开销

结构	GreedyCommLB	对象数目			
		128K	256K	512K	1M
集中式	均衡开销(s)	10.296	21.151	38.104	–
	内存开销(M)	41.405	82.866	165.811	–
层次结构	0 层均衡开销	0.196	0.440	0.782	1.716
	0 层内存开销	5.154	10.328	24.934	41.405
	1 层均衡开销	0.462	1.291	3.694	5.225
	1 层内存开销	10.328	24.934	41.405	82.866

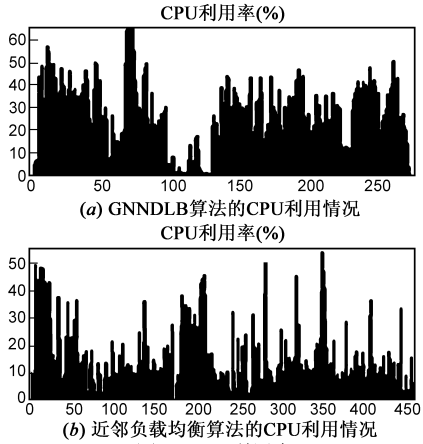
表 2　64K 个模拟处理器上的策略开销

结构	GreedyCommLB	对象数目			
		128K	256K	512K	1M
集中式	均衡开销(s)	10.838	25.217	45.185	–
	内存开销(M)	41.403	82.866	165.811	–
层次结构	0 层均衡开销	0.420	1.082	3.947	4.952
	0 层内存开销	10.328	24.934	41.405	82.866
	1 层均衡开销	0.492	1.032	2.304	5.631
	1 层内存开销	12.453	20.684	41.405	82.866

从表 1 和表 2 可以看出,集中式负载均衡策略中的管理节点存在瓶颈,而层次结构策略通过阈值设定可降低负载均衡树的高度,可大大降低内存开销、叶子节点的空闲时间和负载均衡开销.

3.2　GNNDLB 与传统分布式策略性能比较

为比较 GNNDLB 层次结构策略与传统分布式策略的性能,本文应用 BigSim 仿真器模拟运行了由 64K 个节点和 256K 个对象参与的 GNNDLB 算法与近邻负载均衡算法(将超载任务分配给相邻节点),并用 Projections 工具跟踪负载均衡过程,如图 1(图 1(*a*)和图 1(*b*))所示.



(*a*) GNNDLB算法的CPU利用情况

(*b*) 近邻负载均衡算法的CPU利用情况

图1　CPU利用率

由图 1(*a*)和图 1(*b*)可知,GNNDLB 算法的平均 CPU 利用率在 30%左右,而近邻负载均衡算法的平均 CPU 利用率在 10%左右.

3.3　GNNDLB 与随机延迟策略的性能比较

为了验证层次策略的通讯量预测机制,本文在深腾 1800 集群上使用 2 个节点通过循环运行 CHARM++中的通讯测试程序(Commbench)进行延迟取样(Δt = 1m),测试传输 1048576 字节时 IHA、GNN 和 BP 三种方法预测延迟量的结果.根据预测延迟平方根误差和 IHA 算法遗忘因子 α 的选取之间的关系可知,当 α 取 0.7 时预测效果最好,因而在本文的比较实验中均选取 $\alpha = 0.7$.

为了比较 GNN 和传统 BP 神经网络的收敛时间和学习次数,在权值初始值、训练样本和参数相同的条件下运行这两种方法,得到如表 3 所示的运行结果.

表 3　BP 和 GNN 收敛时间和学习次数比较

算法 \ 数据/组		50	100	150	200
BP	时间(s)	11.875	7.937	3.406	0.281
	步数	156667	63049	15683	771
GNN	时间(s)	0.281	0.296	0.328	0.078
	步数	629	367	225	49

从表 3 可知,BP 的预测精度低于 GNN,收敛时间高于 GNN.在精度较高的情形中,BP 甚至不收敛.为了比较 GNN、IHA 和 BP 算法的预测性能,在系统参数相同的条件下,使用 75 组延迟样本进行训练,预测 100 组数据,得到如图 2 和图 3 所示的预测结果.

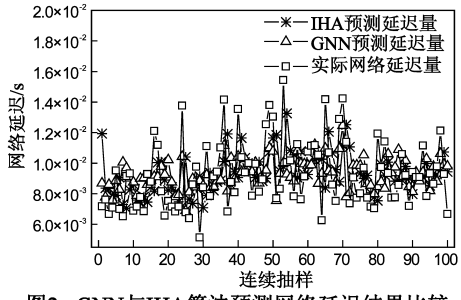


图2　GNN与IHA算法预测网络延迟结果比较

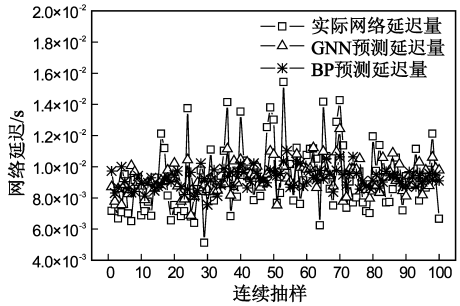


图3　GNN和BP算法预测网络延迟结果比较

从图 2 和图 3 可以看出,GNN 算法的预测结果好于 IHA 和 BP 算法.为了比较这几种方法的预测精度,本文

计算了这几种方法的预测误差,如表 4 所示.

表 4 各种算法的预测误差比较

算法 \ 指标	RME	RMSE	EC	MXARER	MRERR
RW	0.234	0.316	0.851	1.260	0.036
HA	0.206	0.255	0.877	0.898	0.008
IHA	0.197	0.246	0.881	0.907	0.004
BP	0.187	0.230	0.890	0.364	-0.049
GNN	0.165	0.205	0.904	0.471	-0.045

从表 4 可以看出,当延迟变化比较剧烈时,GNN 算法的预测精确度优于 IHA 和 BP 算法.在 GNNDLB 策略均衡过程中,在计算通信成本时,应用预测结果来提高 Makespan 值的精度.从表 5 可知,GNN 预测算法可获得更高的 Makespan 精度.

表 5 GNN 与 IHA 预测 Makespan 值的误差比较

算法 \ 指标	RME	RMSE	EC	MXARER	MRERR
IHA	0.014	0.014	0.992	0.017	0.014
GNN	0.011	0.011	0.994	0.014	0.011

为了考察 GNN 延迟预测方法对负载均衡产生的影响,在一个具有二维网孔结构的分布式系统中,设定各个节点具有 10 个任务,每个任务与其近邻节点的任务通讯 100 次,对基于 GNN 延迟预测方法和 IHA 延迟预测方法的 GreedyCommLB 算法进行性能测试,将所得到的 Makespan 值和实际值进行比较,比较结果如图 4 所示.从图 4 可知,GNN 方法优于 IHA 方法,可提高 Makespan 值的精度.

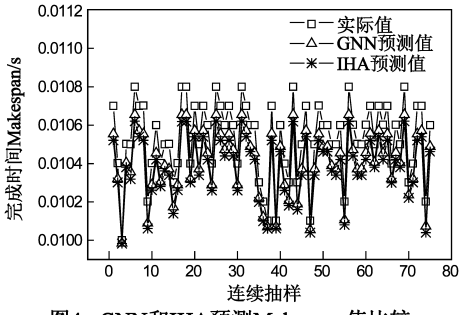


图4 GNN和IHA预测Makespan值比较

4 结论与展望

考虑了大规模并行与分布式计算系统中的负载均衡开销和网络延迟时变性特征,给出一种基于广义神经网络的层次结构负载均衡策略.该策略应用通讯优化的层次结构减少大规模系统的负载均衡开销,可优化任务通讯延迟和迁移延迟.仿真实验验证了本文策略的有效性.

参考文献

[1] Cybenko G. Dynamic load balancing for distributed memory multiprocessors[J]. J Par Distr Comp, 1989, 7(2): 279 - 301.

[2] Hui C C, et al. Hydrodynamic load balancing[J]. IEEE T Parall Distrib, 1999, 10(11): 1118 - 1137.

[3] Casavant T L, et al. A taxonomy of scheduling in general-purpose distributed computing systems[J]. IEEE T Soft Eng, 1988, 14(2): 141 - 154.

[4] Lan Z L, et al. Dynamic load balancing for adaptive mesh refinement application [A]. Proc ICPP'01[C]. Washington: IEEE CS Press, 2001. 571 - 579.

[5] Hu J H, et al. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment [A]. Proc 3rd IEEE PAAP[C]. Los Alamitos: IEEE CS Press, 2010. 89 - 96.

[6] Bhadani A, et al. Performance evaluation of web servers using central load balancing policy over virtual machines on cloud [A]. Proc 3rd ACM COMPUTE[C]. New York: ACM Press, 2010. 1 - 4.

[7] Zhou W Y, et al. VMCTune: A load balancing scheme for virtual machine cluster based on dynamic resource allocation[A]. Proc 9th IEEE GCC[C]. Washington: IEEE CS Press, 2010. 81 - 86.

[8] Vaquero L M, et al. Dynamically scaling applications in the cloud[J]. ACM SIGCOMM Comput Commun Rev, 2011, 41(1): 45 - 52.

[9] Randles M, et al. A comparative study into distributed load balancing algorithms for cloud computing[A]. Proc 24th IEEE WAINA[C]. Washington: IEEE CS Press, 2010. 551 - 556.

[10] Iqbal W, et al. SLA-driven dynamic resource management for multi-tier web applications in a cloud[A]. Proc 10th IEEE/ACM CCGRID[C]. Washington: IEEE CS Press, 2010. 832 - 837.

[11] Kang Y, et al. A user experience-based cloud service redeployment mechanism [A]. Proc 4th IEEE CLOUD[C]. Washington: IEEE CS Press, 2011. 227 - 234.

[12] Oguchi N, et al. Reconfigurable TCP: An architecture for enhanced communication performance in mobile cloud services [A]. Proc. 11th IEEE/IPSJ SAINT[C]. Washington: IEEE CS Press, 2011. 242 - 245.

[13] Martinez J C, et al. Experimental study of large-scale computing on virtualized resources[A]. Proc 3rd ACM VTDC[C]. Barcelona: ACM Press, 2009. 35 - 42.

[14] Song H, et al. PerHPC: Design and implement personal high performance computing platform using cloud computing technology[A]. Proc 6th ChinaGrid[C]. Los Alamitos: IEEE CS Press, 2011. 28 - 34.

[15] Hajjat M, et al. Dealer: Dynamic request splitting for performance-sensitive applications in multicloud environments [R]. Indiana: Purdue University, 2011.

[16] Hayat M M, et al. Dynamic Time Delay Models for Load Bal-

ancing, Part II: A Stochastic Analysis of the Effect of Delay Uncertainty[M]. Berlin: Springer-Verlag, 2004. 355 – 368.

[17] Dhakal S, et al. Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach[J]. IEEE T Parall Distrib, 2007, 18(4): 485 – 497.

[18] Eck J T, et al. An automatic text-free speaker recognition system based on an enhanced art 2 neural architecture[J]. Inform Sci, 1994, 76(3 – 4): 233 – 253.

[19] Park D, et al. Spectral basis neural networks for real-time travel time forecasting[J]. J Transp Eng, 1999, 125(6): 515 – 523.

[20] 胡瑞敏, 等. 广义知识存储原理与高阶广义神经网络[J]. 电子学报, 1996, 24(7): 59 – 65.
Hu R M, et al. Generalized information storing principle and higher-order generalized neural networks[J]. Acta Electronica Sinica, 1996, 24(7): 59 – 65. (in Chinese)

[21] Yagoubi B, et al. Dynamic load balancing strategy for grid computing[J]. Enformatika Transactions on Engineering, Computing and Technology, 2006, 13: 260 – 265.

[22] Parallel Programming Laboratory (PPL). Runtime systems and tools: BigSim-Simulating PetaFLOPS supercomputers [OL]. http://charm.cs.uiuc.edu/research/bigsim, /2012-04-18.

[23] 杨际祥, 等. 并行与分布式计算动态负载均衡策略综述[J]. 电子学报, 2010, 38(5): 1122 – 1130.
Yang J X, et al. A survey of dynamic load balancing strategies for parallel and distributed computing[J]. Acta Electronica Sinica, 2010, 38(5): 1122 – 1130. (in Chinese)

作者简介

杨际祥　男, 博士. 研究方向为并行与分布式计算、优化计算方法与理论及其应用.
E-mail: jixiang_yang@126.com

谭国真　男, 教授, 博士生导师. 研究方向为集群计算、智能交通系统、无线传感器网络与检测技术、VANETs.
E-mail: gztan@dlut.edu.cn