# 基于替换策略的自适应数据库负载控制 *

李海芳, 李彦彪, 强 彦
(太原理工大学 计算机与软件学院, 太原 030024)

**摘 要**: 自适应数据库负载控制是一个性能优化过程,用来提高数据库系统的整体性能。分析了自适应数据库负载控制的研究现状,提出了基于替换策略的负载控制方法,该方法通过竞争决策算法和替换策略对负载进行优化过滤来提高数据库系统的性能。仿真实验表明,该方法可以有效提高客户的满意率和系统实现价值,并且可以增强系统的稳定性。

**关键词**: 自适应数据库; 负载控制; 竞争决策算法; 0/1 背包问题; 替换策略; 优先级表排序
**中图分类号**: TP311.13 **文献标志码**: A **文章编号**: 1001-3695(2010)05-1830-03
doi:10.3969/j.issn.1001-3695.2010.05.063

# Workload control in autonomic database based on replacement strategy

LI Hai-fang, LI Yan-biao, QIANG Yan
(*College of Computer & Software, Taiyuan University of Technology, Taiyuan* 030024, *China*)

**Abstract**: Workload control in autonomic database is a process of performance optimization, which is used to improve the whole performance of database systems. This paper analyzed the research status of workload control in autonomic database and proposed a control method based on replacement strategy, this method carried on the optimization filter through the competitive decision algorithm and replacement strategy to improve the database system performance. The simulation experiments show that this method can improve the customer's satisfaction rate and the system's realization value effectively, it can also improve the system's stability.

**Key words**: autonomic database; workload control; competitive decision algorithm; 0/1-knapsack problem; replacement strategy; priority table ranking

## 0 引言

在信息技术迅速发展的时代,信息量骤增的同时也呈现出信息的复杂性、多变性,仅仅依靠 DBA 根据外在环境变化,通过对系统参数的设置、部署方法的应用等系统调节方法来提高系统的性能,已经不能适应现在大量的、复杂多变的负载了。自适应数据库负载控制是一种新的数据库性能优化技术,在数据库资源一定的条件下,通过区分不同的访问需求,数据库系统智能地决定先响应哪些访问,再响应哪些访问,动态合理地规划负载的访问顺序来更合理地满足对数据库系统的访问需求[1],其通过对负载本身进行优化过滤来提高系统性能。

文献[2]通过在数据库内部为不同的负载分配不同的资源来提高系统的性能,由于需要在数据库内部优化,实现起来比较困难且不灵活;文献[3]在数据库外部进行负载过滤,通过设定 MPL(multi-programming limit)值来确定要提交的负载数量,虽然实现起来比较灵活,但是要确定一个合适的 MPL 值并不容易;文献[4]只是根据系统当前空闲资源,按照先来先服务的原则来控制负载数量,该方法只是用来提高系统的稳定性,没有考虑系统的其他性能。

本文综合考虑了负载的各种特征(负载价值、负载所需消耗 CPU 资源、空闲时间),应用替换策略进行负载控制,具体的控制过程分为两步:首先应用搜索优化算法,在系统 CPU 资源允许的条件下,寻找一组价值最大的负载作为预提交对象,这一过程称之为搜索优化过程;第二步是在剩余的负载中用那些紧迫性很强的负载替换预提交对象中价值密度较小且空闲时间较长的负载,这一过程用优先级表排序方法来实现,称之为替换过程。该方法可以灵活有效地控制各种复杂多变的负载,既可以提高客户的满意度,又可以提高数据库的稳定性和实现价值。

## 1 用竞争决策算法求价值最大负载

在系统 CPU 资源允许的条件下,寻找一组价值最大的负载作为预提交对象,这是一个典型的 0/1 背包问题[5]。目前有许多求解 0/1 背包问题的算法,主要分为精确算法和近似算法。精确算法有如动态规划、回溯法等;近似算法有贪心算法、遗传算法、蚁群算法等[6]。精确算法虽然可以找到问题的最优解,但是所用的时间和空间代价太大,而近似算法中的贪心算法虽然可以在短时间内找到算法的近似解,但贪心算法容易陷入局部最优,所求结果不太理想,而粒群算法、蚁群算法等近似算法的收敛速度较慢[7]。不适合应用在自适应数据库负载控制中。

上海理工大学的宁爱兵博士提出一种竞争决策算法(competitive decision algorithm, CDA)来求解 0/1 背包问题[7]。

**收稿日期**: 2009-09-28; **修回日期**: 2009-11-05 **基金项目**: 国家自然科学基金资助项目(60970059); 山西省自然科学基金资助项目(2007011050)

**作者简介**:李海芳(1963-),女,山西昔阳人,教授,博士,主要研究方向为智能信息处理、数据库负载自适应技术;李彦彪(1984-),男,硕士研究生,主要研究方向为数据库负载自适应技术(liyanbiao0113@163.com);强彦(1969-),男,博士研究生,主要研究方向为数据库负载自适应技术.

该算法在贪心算法的基础上通过竞争决策机制对搜索结果进行优化,使搜索结果尽可能得到最优解,而且时间复杂度小于其他近似算法。

### 1.1　问题描述

1)负载描述

假设有负载 $W_i$,首先给出负载特征参数的定义:

a)$a_i$ 表示负载到达时间,即控制器将负载拦截下来的时刻。

b)$t_i$ 表示当前时刻。

c)$c_i$ 表示客户可以容忍的负载最大响应时间,即如果负载在该时间内能够得到响应,则表示客户对服务满意,超过该时间则不满意。

d)$e_i$ 表示系统响应该负载所需要的时间,即从负载被提交给数据库系统到返回处理结果所需要的时间。

e)$s_i$ 表示空闲时间,该时间用来衡量负载的紧迫性,即 $s_i = c_i - (t_i - a_i) - e_i$。

f)$v_i$ 表示负载的价值,即该负载相对于其他负载的重要程度。

g)$\mathrm{cpu}_i$ 表示数据库系统执行该负载所需消耗 CPU 的资源。

h)$vd_i$ 表示负载的价值密度,即单位时间内负载的价值,$vd_i = v_i / \mathrm{cpu}_i$。

可以将负载 $W_i$ 用三元组表示为 $W_i = (v_i, \mathrm{cpu}_i, s_i)$。

2)0/1 背包问题定义

0/1 背包问题可以描述如下:已知 $n$ 个物品的重量和价值分别为 $w_i > 0$ 和 $p_i > 0 (i = 1,2,\cdots,n)$,背包能承担的最大重量为 $C > 0$,要求在总重量不超过 $C$ 的条件下,选择一组总价值最大的物品装入背包,该问题的数学模型可以表示为[8]

$$\begin{cases} \max \quad f(x_1,x_2,\cdots,x_n) = \sum_{i=1}^{n} p_i x_i \\ \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leqslant C \\ \qquad x_i \in \{0,1\} \quad (i = 1,2,\cdots,n) \end{cases} \tag{1}$$

其中:$x_i$ 为 $\{0,1\}$ 决策变量,$x_i = 1$ 表示将物品 $i$ 装入背包中,$x_i = 0$ 则表示不将其装入。

在自适应数据库负载控制中,将数据库系统看做背包,负载 $W_i$ 看做物品,则负载价值 $v_i$ 为物品 $i$ 的价值、$\mathrm{cpu}_i$ 为物品 $i$ 的负重,系统当前空闲的 CPU 资源为背包能承担的最大负重。搜索优化过程的目的是在满足条件 $\sum_{i=1}^{n} \mathrm{cpu}_i x_i \leqslant V$ 的情况下求得 $\sum_{i=1}^{n} v_i x_i$ 最大值。其中 $V$ 表示系统当前空闲的 CPU 资源。

### 1.2　竞争决策算法原理

竞争决策算法是通过构造一个或多个竞争者参与到对一个或多个资源的竞争过程中,通过优胜劣汰的决策原则使一部分竞争者获得资源而增加实力,使另一部分竞争者失去资源而减弱实力甚至死亡[7]。

在自适应数据库负载控制中,把负载作为各竞争者要争夺的资源 ,数据库系统当做竞争者 $A$,另一个竞争者是虚拟竞争者 $N$,竞争开始时,虚拟竞争者 $N$ 占有所有负载, 竞争者 $A$ 不占有任何负载,竞争结束时竞争者 $A$ 占有的负载就是要提交给数据库的负载。

1)基本符号及含义

$s_0$ 表示还未装入到背包中的负载队列,$s_0$ 按价值密度 $vd_i$ 从大到小排序。

$s_1$ 表示已装入到背包中的负载队列,$s_1$ 按价值密度 $vd_i$ 从小到大排序。

$\mathrm{cpu}_f$ 表示背包在装入 $s_1$ 中的所有负载后剩余的 cpu 资源,即有

$$\mathrm{cpu}_f = V - \sum_{i \in s_1} \mathrm{cpu}_i \tag{2}$$

power[$i$]为竞争者 $A$ 对当前队列 $s_0$ 中的负载 $i$ 的竞争力函数。

2)算法描述

竞争决策算法描述如下[7]:

a)初始状态。所有负载在最开始时全部被虚拟竞争者 $N$ 占有,竞争者 $A$ 没有占用任何负载。

b)竞争力函数。

$$\mathrm{power}[i] = \begin{cases} vd_i & i \in s_0 \ \&\ \&\ \mathrm{cpu}_i \leqslant \mathrm{cpu}_f \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

c)决策函数。若存在竞争力函数值大于 0 的负载,则按照竞争力函数把符合要求的负载 $k$ 分配给竞争者 $A$,即把竞争力函数值最大的负载装入到背包中并按如下方式修改:$s_0 = s_0 - \{k\}$,$s_1 = s_1 + \{k\}$,$v_f = v_f - \mathrm{cpu}_k$,重新计算竞争力并转 c),直到没有竞争力函数值大于 0 的负载为止。

d)资源交换规则。交换竞争者 $A$ 与虚拟竞争者 $N$ 之间的一些负载,直到不能再通过负载交换得到更好的解为止,具体操作步骤如下:

(a)$j = 1$,$m$ 为 $s_1$ 中的负载的个数;

(b)假设 $s_1$ 中第 $j$ 个负载的编号为 $k$,令 $\mathrm{cpu}_r = \mathrm{cpu}_f + \mathrm{cpu}_k$;此时计算把 $s_0$ 中的负载装入到负重为 $\mathrm{cpu}_r$ 的背包中能取得的总价值 $v_s$,计算过程中采用先把价值密度大且能装入的负载先装入的方法,计算过程中装入的物品集合为 $s_3$,$s_3$ 中的物品价值总和为 $v_s$,$s_3$ 中的负载 cpu 消耗总和为 $\mathrm{cpu}_s$;

(c)若 $v_s > v_k$,则有集合 $s_0 = s_0 + \{k\} - s_3$,$s_1 = s_1 - \{k\} + s_3$,$\mathrm{cpu}_f = \mathrm{cpu}_r - \mathrm{cpu}_s$ 在背包中把负载 $k$ 移出,而把 $s_3$ 中的所有负载装入到背包中,跳到(a)继续;

(d)$j = j + 1$;

(e)若 $j \leqslant m$,则跳到(b)继续,否则算法结束,$s_1$ 中的负载就是所求的预提交负载对象。

文献[7]已证明用竞争决策算法求解 0/1 背包问题,不论是求解速度还是求解精度方面都有不俗的表现。

## 2　基于优先级表的负载替换策略

在数据库过载的情况下,为了既可以提高客户的满意率,又可以提高数据库实现价值,需要将那些在集合 $s_0$ 中紧迫性很强的负载替换进预提交对象集合 $s_1$ 中。具体方法是将集合 $s_1$ 中的负载按照价值密度 $vd_i$ 和空闲时间 $s_i$ 进行综合排序,使得价值密度较小且空闲时间较大的负载能被集合 $s_0$ 中紧迫性很强的负载优先替换出去,这样既可以提高客户的满意率,又可以提高数据库实现的价值。

由于负载的价值密度 $vd_i$ 和空闲时间 $s_i$ 的性质完全不同,不能对其进行简单的算术运算,在这里使用文献[9]提出的优先级表综合排序算法对 $s_1$ 中的负载进行综合排序。

### 2.1　优先级表排序算法

基于优先级表的综合排序法如下:首先对 $s_1$ 中的负载分别按照价值密度 $vd_i$ 进行升序排序,按照空闲时间进行降序排

序,$i$,$j$ 分别为负载在价值密度队列和空闲时间队列中的位置。

这样每个负载都有一个优先等级 $P$[9]:

$$P = i + j \tag{4}$$

优先等级高的负载(指那些价值密度小且空闲时间长的负载)优先被替换出去,在这里由于不同的负载可能有相同的优先等级,比如说,负载 1 在价值密度队列和空闲时间队列中的位置分别为 1 和 3,负载 2 在价值密度队列和空闲时间队列中的位置分别为 3 和 1,则负载 1 和 2 的优先等级相等,均为 4,所以需要为每个负载分配惟一的优先级 $p$,本文用文献[9]中的优先级分配公式为每个负载分配惟一的优先级 $p$:

$$p = (i + j - 1) * (i + j - 2)/2 + i \tag{5}$$

根据优先级函数 $P$ 就可以将 $s_1$ 中那些价值密度较小且空闲时间较长的负载排到前边,作为优先被替换的对象。

### 2.2　替换策略

首先对预提交的负载集合 $s_1$ 中的负载按照优先级公式 $p$ 进行排序,使得那些价值密度相对较小且空闲时间较长的负载排到前边作为优先提交对象,然后进行替换操作,具体的替换策略如下:

a)在剩余负载集合 $s_0$ 中找出紧迫性较强的负载放入集合 $s_4$ 中,假设紧迫性较强的负载个数为 $n$,集合 $s_5$ 用来存放集合 $s_1$ 中要被替换出的负载。

b)在 $s_4$ 中第 $i(i = 1,2,\cdots,n)$ 个负载的编号为 $k$,此时计算把 $s_1$ 中的负载装入总负重为 $\mathrm{cpu}_k$ 的背包中,计算时将那些价值较小且空闲时间较长的负载优先装入背包,并将要被替换出的负载放入集合 $s_5$ 中,具体替换步骤如下:

步骤 1 设置初始值 $j = 1$,$\mathrm{cpu}_t = \mathrm{cpu}_f$,$s_1^j$ 表示集合 $s_1$ 中第 $j$ 个负载的标志符。

步骤 2 令

$$\mathrm{cpu}_t = \mathrm{cpu}_t + \mathrm{cpu}_{s_1^j} \tag{6}$$

若 $\mathrm{cpu}_t < \mathrm{cpu}_k$,则 $s_5 = s_5 + \{s_1^j\}$,$j = j + 1$,否则跳到步骤 3;

步骤 3 对集合 $s_1$ 进行替换操作,即 $s_1 = s_1 + \{k\} - s_5$,并标记负载 $k$ 不能被替换。

c)清空集合 $s_5$,为替换集合 $s_4$ 中的下一个负载做准备,依此类推,直至将集合 $s_4$ 中的所有负载都替换进集合 $s_1$ 为止,此时的集合 $s_1$ 为最终提交给数据库系统的负载集合。

应用替换策略可以有效地将集合 $s_0$ 中紧迫性很强的负载替换进入集合 $s_1$ 中,作为优先提交对象提交给数据库系统,该方法在提高系统实现价值的同时提高了客户的满意率。

## 3　实验结果分析

根据实际需要,本文将系统实现价值和用户满意率作为衡量自适应数据库负载控制性能的依据,仿真实验也是围绕这两个性能评价参数进行的,其中用户满意率定义如下:

$\mathrm{count}_i$ 表示在第 $i$ 个控制时间间隔内负载的总个数;

$\mathrm{no\_count}_i$ 表示在第 $i$ 个控制时间间隔内没有被提交的且空闲时间为 1 的负载的个数;

sat_rat 表示用户满意率,则有:

$$\mathrm{sat\_rat}_i = \mathrm{no\_count}_i / \mathrm{count}_i \tag{7}$$

在实验过程中,负载的价值是由随机函数随机生成的 1 ~ 10 的任意值;负载所需消耗的 CPU 值是由随机函数随机生成的 1% ~40% 的任意值;空闲时间是由随机函数随机生成的 1 ~10(单位为 s)的任意值。实验进行了 20 个控制间隔,每个控制间隔时间为 1 s。实验结果如图 1、2 所示。

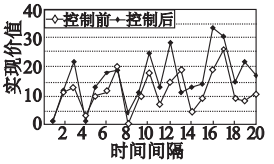


图1　系统实现价值

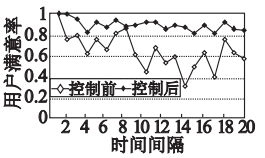


图2　用户满意率

由图 1 表明,在绝大多数控制时间间隔内,控制后的系统实现价值要比控制前的系统实现价值高。由表 1 可以得出,在 20 个控制时间间隔内,控制后系统实现的总价值比控制前提高了 45.3%;由图 2 表明,控制前用户的满意率在 34% ~ 100% 之间波动,控制后用户的满意率保持在 80% 以上,由表 1 可知控制后的平均用户满意率比控制前的平均用户满意率提高了 22%。实验结果表明,基于替换策略的负载控制方法不仅可以提高系统的实现价值,使得那些价值密度较大的负载能够优先执行,而且有效地提高了用户的满意度。

表 1　控制前后实验数据对比

| 对比项 | 系统实现总价值 | 平均用户满意率/% |
|---|---|---|
| 控制前 | 223 | 66.3 |
| 控制后 | 325 | 90.2 |

## 4　结束语

本文应用替换策略进行自适应数据库负载控制,首先采用竞争决策算法求解一组价值最大的负载作为预提交对象,然后应用替换策略将剩余负载中那些紧迫性很强的负载替换进入提交队列。该方法可以对各类复杂多变的负载进行有效的优化过滤,在实现系统价值最大化的同时有效地提高了客户的满意率,达到了优化数据库性能的目的。本文只考虑了对系统 CPU 资源的利用,没有考虑内存等其他系统资源,在以后的研究工作中可以在这一方面进行尝试。

**参考文献:**

[1] MANEVITZ L M, YOUSEF M. One-class SVMs for document classification [J]. Journal of Machine Learning Research, 2001, 2(22): 139-154.

[2] NIU Bao-ning, MARTIN P, POWLEY W, *et al*. Workload adaptation in autonomic DBMSs[C]// Proc of CASCON. Toronto: ACM Press, 2006: 161-173.

[3] SCHROEDER B, HARCHOL-BALTER M, IYENGAR A, *et al*. Achieving class-based QoS for transactional workloads[C]// Proc of the 22nd International Conference on Data Engineering (ICDE'06). Washington DC: IEEE Computer Society, 2006: 153-165.

[4] DUAN Fu, WANG Yu-xing, ZHAO Chan-chan, *et al*. Research and implementation on middleware of database workload autonomic adaptation[C]// Proc of the 4th International Conference on Intelligent Information Hiding and Multimedia Signal Processing. Washington DC: IEEE Computer Society, 2008: 1478-1481.

[5] QIANG Yan, LI Yi, CHEN Jun-jie. The workload adaptation in autonomic DBMSs based on layered queuing network model[C]// Proc of the 2nd International Workshop on Knowledge Discovery and Date Mining. Washington DC: IEEE Computer Society, 2009:781-785.

[6] 强彦,李晶,陈俊杰.数据库负载自适应的体系结构研究[J].计算机应用研究,2008, 25(11): 3317-3319.

[7] 宁爱兵,马良.0/1 背包问题竞争决策算法 [J].计算机工程与应用,2008,44(3):14-16.

[8] 周明,孙树栋.遗传算法原理及应用[M].北京:国防工业出版社,1996.

[9] 王强,徐俊刚.一种新的基于优先级表的实时调度算法[J].电子学报,2004,32(2):310-313.