第16卷 第4期
2006年4月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.4
Apr. 2006

模拟退火算法在全局查询优化中的应用

林慧君[1],彭 宏[2]
(1.湛江教育学院 计算机科学系,广东 湛江 524037;
2.华南理工大学 计算机科学与工程学院,广东 广州 510641)

摘 要:在分布式环境下,全局查询的代价函数空间形状包含了很多局部最小状态,需要多次局部最优化才可以找到全局最小状态。模拟退火算法是目前发展较快的智能优化算法,是一种以概率1收敛于全局最优解的全局优化算法。文中讨论了全局查询优化的过程以及模拟退火算法在全局查询优化中的应用,并对算法进行了一些改进。

关键词:全局查询;优化;模拟退火算法;代价函数;
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1005-3751(2006)04-0155-03

Application of Simulated Annealing in Global Query Optimization

LIN Hui-jun[1],PENG Hong[2]
(1.Computer Science Department of Zhanjiang Educational College,Zhanjiang 524037,China;
2. Computer Sci. and Eng. College of South China Univ. of Techn., Guangzhou 510641,China)

Abstract: In the distributed environment, the cost function space shape of global query includes many partial minimum status, so need to do partial optimization for some times in oder to find the global minimum status . Simulated annealing is an intelligent algorithm of developing very fast. In this paper, discuss some process of global query optimization and the application of simulated annealing.

Key words: global query; optimization; simulated annealing; cost function

0 引 言

在数据库应用系统中,查询是指从数据库中提取数据的一系列活动[1]。在查询的过程中,为了减少查询的处理时间,提高系统的处理能力,需要对查询的处理过程进行优化处理。查询的优化就是从说明性查询语言生成最优查询执行计划的过程[1]。传统查询优化的方法比较多,如代数优化、物理优化、规则优化、代价估算优化等等。

在分布式环境中,由于数据是分布在不同地理位置的各个数据节点上,并由各数据节点独立进行管理,这给数据的查询优化过程带来了很多问题,如数据的收集缺乏足够的和精确的统计数据,并需要消耗系统大量的资源,查询代价的估计与查询具有的连接数量成指数级增长,无法在查询执行前准确地估计查询执行代价等等。这使得传统的一次优化一次执行的查询优化方法不适应分布式环境的查询优化过程。

模拟退火算法是目前发展较快的全局反演算法,具有渐近收敛性,已在理论上被证明是一种以概率1收敛于全局最优解的全局优化算法[2]。文中讨论了该算法在分布式环境下全局查询优化中的应用。

收稿日期:2005-10-14
基金项目:广东省科技攻关项目(A10202001)
作者简介:林慧君(1971-),女,广东湛江人,硕士,研究方向为网络和数据库;彭 宏,教授,博士生导师,研究方向为人工智能。

1 全局查询优化的过程

全局查询优化是为一个给定的全局查询语言选择一个优化了的查询执行计划的过程。如图1所示。

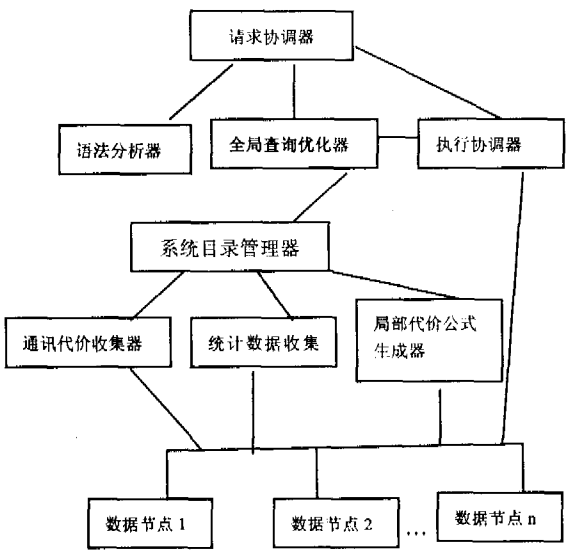

图1 全局查询结构图

当用户发出一个查询请求时,请求协调器调用语法分析器进行语法分析,并生成查询的分析树,请求协调器将查询分析树传至全局查询优化器进行优化处理后得到一个最优的查询执行计划,调用执行协调器执行该查询计划,并将查询结果由请求协调器传给用户。

全局查询优化的第一步是要生成查询处理树，并找出其最优的查询执行计划，经典的查询处理树生成算法是 System－R[3] 使用的动态程序法，该算法使用等价规则系统产生与给定表达式等价的全部表达式，然后生成相应的执行计划，最后通过估算各执行计划的代价选择一个代价最小的执行计划，算法的缺点是当可选的执行计划很多时，需要消耗大量的计算时间。文中使用模拟退火算法替代动态程序法以寻求最优的查询执行计划。

在全局查询优化中，首先使用目录管理器收集各种代价信息，包括通讯代价信息、统计数据信息等，并使用局部代价公式生成器生成局部代价信息，然后定义一个搜索的空间，空间中的每一个点就是一个状态，对应一个待优化查询的查询处理树，每个状态都有一个代价与之相关，这个代价是由代价函数给出，优化算法的目标是找出状态空间内代价最小的状态。

当状态空间越大，所包括的状态越多，为了减少状态空间，引入启发式规则[4]，启发式规则是尽早地执行选择和投影操作。当应用了启发式规则后，只要确定了连接的次序，就可以确定选择和投影操作执行的位置，因此优化的目标改为查找最优的连接次序、连接方法以及连接的执行地点，其连接处理树是以叶子节点为关系基表，中间节点为连接操作算子，边为数据流的树。

2 查询优化的代价函数

在分布式环境下，查询计划执行的代价主要由通讯的代价与局部操作代价构成，其中局部操作代价包括访问辅助存储器的代价，即 I/O 代价和计算代价。

2.1 I/O 代价的估算模型

访问磁盘一次所需用的代价表示为[5]：

$$C_{I/O} = D_0 + D_1 * x$$

其中，x 是存取数据的大小，以字节数表示；D_0 是与 x 无关的 I/O 代价，包括寻道时间和等待时间；D_1 是每个字节所需的传输时间。

一般来说，$D_0 >> D_1 * x$，故 $C_{I/O} \approx D_0$

loc_cost(OP) = OP 的 I/O 次数 * D_0

2.2 通讯代价估算模型

一次的通讯代价表示为 $C_C(x) = C_0 + C_1 * x$

其中，x 为传送数据的大小，以字节数表示；C_0 是传送一次数据所必需的初始代价；C_1 是数据传送的速率。

一般来说，$C_0 << C_1 * x$，因此，$C_C(x) \approx C_1 * x$

comm_cost(OP) = $C_1 * x$(OP)

其中，x(OP) 为执行 OP 时所需传送数据的大小。

2.3 查询的代价函数

一个查询执行计划的代价表示为：

cost = $W + R$

其中，W 是以时间表示的查询所消耗的总工作量，R 是查询的总响应时间。

在综合考虑通讯代价和局部操作代价时，进行适当加权：

$$W = a_{twcc} * W_c + a_{twlp} * W_L$$

$$R = a_{rtcc} * R_c + a_{rtlp} * R_L$$

其中，W_c 是以时间表示的通讯代价总工作量，W_L 是以时间表示的局部操作代价总工作量，R_c 是以时间表示的通讯代价响应时间，R_L 是以时间表示的局部操作代价响应时间。这样总代价公式为：

$$\text{cost} = a_{twcc} * W_c + a_{twlp} * W_L + a_{rtcc} * R_c + a_{rtlp} * R_L$$

其中 cost，W，R，W_c，W_L，R_c，R_L 都是以查询执行计划为参数的函数。

记查询执行计划为 Q，其顶点操作算子为 OP，两个子树分别为 Q_{left}、Q_{right}，则

$$Q = Q_{left} \text{OP } Q_{right}$$

Q 的代价值可递归计算如下：

$$W_c(Q) = W_c(Q_{left}) + W_c(Q_{right}) + \text{comm\_cost(OP)}$$

$$W_l(Q) = W_l(Q_{left}) + W_l(Q_{right}) + \text{loc\_cost(OP)}$$

$$R_C(Q) = \text{MAX}[R_c(Q_{left}), R_c(Q_{right})] + \text{comm\_cost(OP)}$$

$$R_L(Q) = \text{MAX}[R_L(Q_{left}), R_L(Q_{right})] + \text{loc\_cost(OP)}$$

其中 comm_cost(OP)是将子树输出结果传送到 OP 执行节点的通讯代价，loc_cost(OP)是执行操作算子 OP 的局部操作代价。

3 模拟退火算法

模拟退火算法的概念源于统计物理学，是模拟固体熔化状态逐渐缓慢冷却后最终达到结晶状态这一物理过程。其基本的原理是：将固体加温至充分高，再让其徐徐冷却。加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 $e^{-\Delta E/(kT)}$，其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。

在全局查询的状态空间中，一个状态对应一个连接处理树，模拟退火算法就是在状态空间内进行一系列的状态转换操作，以寻找代价最小的状态。

设 A，B，C 为任何的连接处理公式，∞为该连接方法，则状态转换的规则表示如下：

1)连接方法选择：$A\infty_{method_i}B \leftrightarrow A\infty_{method_j}B$(改变连接操作算子的的执行方法)

2)连接执行地点选择：$A\infty_{site_m}B \leftrightarrow A\infty_{site_n}B$

3)连接交换律：$A\infty B \rightarrow B\infty A$

4)连接结合律：$(A\infty B)\infty C \leftrightarrow A\infty(B\infty C)$

5)左连接交换律：$(A\infty B)\infty C \rightarrow (A\infty C)\infty B$

6)右连接交换律：$A\infty(B\infty C) \rightarrow B\infty(A\infty C)$

根据上述规则，设状态空间为 M，S_0 为状态空间的一个初始状态，cost(S)为 S 的代价，neihbors(S)为状态 S 的邻居状态集，neihbors(S) 是使用上述规则进行状态转

换后得到的状态集合,则查询处理树的模拟退火算法如下:

```
procedure CXSA(){
    S = S0;
    T = T0;
    minS = S;
    while not(frozen) do{
        while not(equibrium) do{
S' = random state in neighbors( S );
C = cost( S' ) - cost( S );
If ( C <=0) then S = S' ;
If ( C >0) then S = S' with probability e- C/T;
If cost( S )<cost(minS ) then minS = S; }
T = reduce( T );}
return(minS );}
```

4 算法的改进

当操作算子连接的数目不少于 2 时,代价函数的空间形状如图 2 所示。

图 2　代价函数的空间形状

从图 2 上看,代价函数空间形状属于一个"杯"形,在杯底区域包含了很多局部最小状态,需要多次局部最优化才可以找到全局最小状态。在使用模拟退火算法进行优化时,能在杯底较快地找到全局的最小状态。但是温度 T 的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一,如果初始代价选择在高代价区域,则搜索到全局最优解的可能性大,但是由于初始温度较高,要消耗一定的时间才可以稳定在杯底区域,因此要花费大量的计算时间,使得查询优化的效率较低。因此笔者对算法做出如下改进:

1)改进温度 T 的初始值设置方法。

在确定模拟退火算法的初始代价时,先经过一个重复的局部优化,找出一个较优的局部最小状态,并以此作为模拟退火算法的初值。

算法如下:

```
procedure jbcx()
{minS = S0;
while not (stopping_condition) do{
    s = random state;
    while not (local_minium( s ))do{
        S' = random state in neighbors( S );
If cost( S' )<cost( S ) then S = S' ;
    }
if cost( S' )<cost( S ) then minS = S' }
S0 =min( S );
    }
```

得到的 S_0 是局部最小优值,将其送至 CXSA 进行全局最优处理。这样可以加快优化的速度。

(2) 改进温度管理问题。

考虑计算复杂度的切实可行性等问题,采用如下所示的降温方式:

$$T(t+1) = k \times T(t)$$

式中 k 为正的略小于 1.00 的常数,t 为降温的次数。

5 小　结

在分布式环境下,由于数据库的分布性和自治性给数据的查询优化带来了很多不便,传统的查询优化方法不适合分布式环境下的查询优化。文中分析了分布式环境下全局查询优化的过程,并使用模拟退火算法去解决该过程。模拟退火算法具有质量高、初始鲁棒性强、通用易实现的优点[2],但需要较长的计算时间,因此文中还提出了一种改进算法,通过改进模拟退火算法的初始值设置方法以及改进降温方法提高其计算的速度。

参考文献:

[1] Silberschatz A, Korth H F, Sudarshan S. 数据库系统概念[M]. 杨冬青,唐世渭,等译. 北京:机械工业出版社,2002.

[2] 熊平华. 基于网格计算平台的智能优化算法应用与研究[D]. 杭州:浙江大学,2004.

[3] Steinbrunn M, Moerkotte G, Kemper A. Heuristic and Randomized Optimization for the Join Ordering Problem[J]. The VLDB Journal, 1997, 6(3): 8 - 17.

[4] Selinger P G, Astrahan M M, Chamberlin D D, et al. Acess Path Selection in a Relational Database Management System [A]. In Processding of the ACM SIGMOD Conf. on Management of Data[C]. Boston, USA: [s. n.], 1979. 8 - 17.

[5] 王能斌. 数据库系统原理[M]. 北京:电子工业出版社, 2000. 116 - 135.

(上接第 154 页)

[3] Carnegie Mellon University and IBM. AFS File System[EB/OL]. http://www.cs.wisc.edu/csl/doc/howto/afs, 1995.

[4] Wahl M. Lightweight Directory Access Protocol (v3)[S]. RFC2251. 1997 - 12.

[5] OGSA 结构描述, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft 2.9, 6/22/2002)[EB/OL]. http://www.gridforum.org/ogsi - wg/drafts/ogsa - drafts2. 9-2002 - 06 - 22. pdf. 2002 - 06.