优化处理并行数据库查询的并行数据流方法*

李建中

(黑龙江大学信息技术研究所 哈尔滨 150080)

摘要 本文使用并行数据流技术优化和处理并行数据库查询的方法,提出了一整套相关算法,并给出了一个基于并行数据流方法的并行数据库查询优化处理器的完整设计.这些算法和相应的查询优化处理器已经用于作者自行设计的并行数据库管理系统原型.实践证明,并行数据流方法不仅能够快速有效地实现并行数据库管理系统,也能够有效地进行并行数据库查询的优化处理.

关键词 并行数据库,查询优化,并行数据流,并行查询优化.

中图法分类号 TP311.13

并行数据库查询优化问题不同于传统数据库查询优化问题,需要新的理论、技术和方法.目前已经出现很多优化处理并行数据库查询的方法.[1,2]这些方法分为两类.第1类是两阶段优化方法[3,4],即首先构造优化的顺序查询执行计划,然后并行化该计划,实现查询的并行执行.第2类是单阶段优化方法[5~9],直接产生优化的并行查询执行计划.本文提出了一种使用并行数据流技术的两阶段优化方法.第1阶段使用传统方法构造优化的顺序查询执行计划,第2阶段用并行数据流技术使之并行化.并行数据流方法的初衷是试图在现有数据库理论、技术和方法的基础上实现并行数据库系统.[10,11]但是,到目前为止,这方面的研究很不深入,很少考虑优化问题.本文对这种方法进行了深入系统的研究,充分考虑了优化问题,提出了一整套相关算法,给出了一个基于这种方法的并行数据库查询优化处理器的完整设计.本文提出的算法和查询优化处理器已经用于作者自行设计的并行数据库管理系统原型.实践证明,并行数据流方法不仅能够快速有效地实现并行数据库管理系统,也能够高效率地优化和处理并行数据库查询.

1 并行数据流图

定义 1.1. 设 Q 是一个查询,Q 的简单并行数据流图是一个加权有向树 $T=(\langle V,E\rangle,W,F)$,简称 SPDF(simple parallel dataflow)图,其中 $\langle V,E\rangle$ 是有向树,V 是 Q 中所有操作和关系的集合,V 的每个内结点表示一个操作,每个叶结点是一个关系,如果 v_1 的计算结果是 v_2 的操作量,则存在一条从 v_1 到 v_2 的有向边 $\langle v_1,v_2\rangle\in E$. W 是权集合 $\{p,s\}$,F 是由 E 到 W 的加权函数.

在一个 SPDF 图中,若边 $\langle v_1,v_2\rangle$ 具有权 p,则 v_1 与 v_2 可以按流水线方式并行执行;如果边 $\langle v_1,v_2\rangle$ 具有权 s,则 v_1 和 v_2 必须按照先 v_1 后 v_2 的顺序执行.若 $P_1=(v_1,\ldots,v_k)$ 和 $P_2=(w_1,\ldots,w_m)$ 是 SPDF 图中从 v_1 到 v_k 和从 w_1 到 w_m 的两条有向路径,而且对于 $1\leqslant i\leqslant k$,$1\leqslant j\leqslant m$,$v_i\neq w_j$,则 P_1 上的任何结点和 P_2 上的任何结点都可以并行执行.图1给出了一个 SQL 查询及其 SPDF 图.图中结点"Scan"表示扫描关系的操作,可视为选择、投影或选择与投影的组合操作.以后,我们用叶结点同时表示关系和该关系上的 Scan 操作.

SPDF 图仅能表示操作间的并行性,不能表示单个操作内的并行性.我们需扩展 SPDF 图.

定义 1.2. merge 和 split 是如下定义的两个操作:

(1) merge(n):把 n 个输入数据流合并为一个输出数据流.

(2) split(n):把输入数据流分解为 n 个输出数据流.

定义 1.3. 设 Q 是一个查询,其 SPDF 图是 DF_Q.Q 的复杂并行数据流图(简记 CPDF(complex parallel dataflow)图)由 DF_Q 如下产生:对于 DF_Q 中每条边 (α,β),

(1) 把 α 划分为 k 个结点 $\alpha_1,\ldots,\alpha_k$,把 β 划分为 m 个结点 $\beta_1,\ldots,\beta_m$,$k,m\geqslant 1$,α 和所有 α_i 表示相同的操作,β 和

---

* 本文研究得到国家自然科学基金、国家863高科技项目基金、国家杰出青年基金和黑龙江省杰出青年基金资助.作者李建中,1950年生,教授,博士导师,主要研究领域为数据库与并行计算.

本文通讯联系人:李建中,哈尔滨 150080,黑龙江大学信息技术研究所

本文 1997-03-01 收到原稿,1997-07-25 收到修改稿

© 中国科学院软件研究所 http://www.jos.org.cn

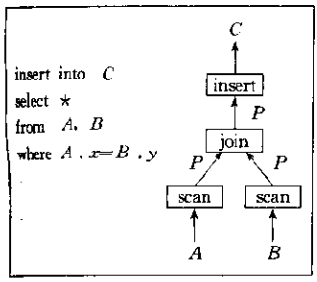

图1　一个SQL查询语句及其SPDF图

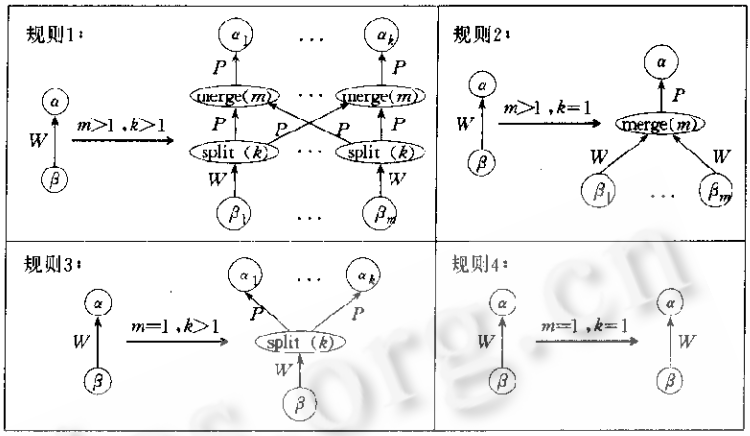

图2　SPDF图的边变换规则

所有β_j表示相同的操作；

(2) 根据不同的m和k值，按照图2所示的规则把边(α,β)变换为一个子图.

定义1.3(1)中的k和m是结点α和β的并行度，表示α(或β)可以由k(或m)个处理机并行执行.CPDF图能够表示关系数据库查询的所有3种并行性.[1,2]

下面以图1中的SQL查询语句为例，说明如何使用CPDF图实现查询的并行处理.设关系A已经划分为子集$A0$,$A1$和$A2$,存放在3个处理机上；关系B已经划分为子集$B0$和$B1$,存放在两个处理机上；关系C已经划分为子集$C0$,$C1$和$C2$,存放在3个处理机上.我们把A上的SCAN操作划分为3个SCAN操作SCAN1,SCAN2和SCAN3；把B上的SCAN操作划分为两个SCAN操作：SCAN4和SCAN5；把JOIN操作划分为3个JOIN操作：JOIN1,JOIN2和JOIN3；把INSERT操作划分为3个INSERT操作：INSERT1,INSERT2和INSERT3.结果CPDF图如图3所示.

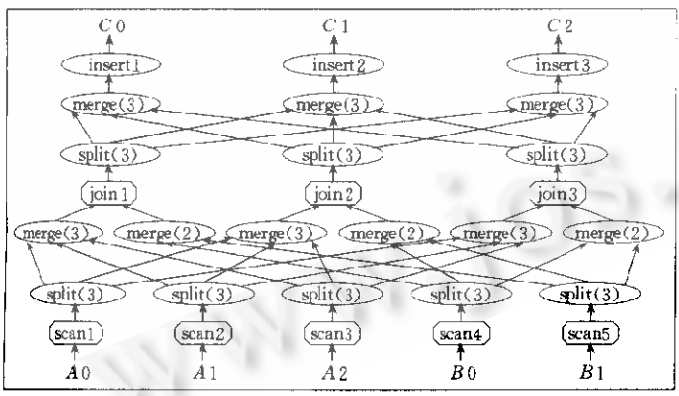

图3　图1中SQL查询的CPDF图(省略了边的权值)

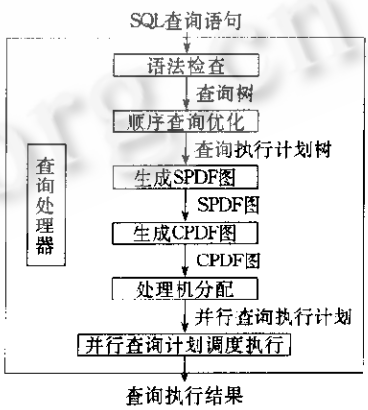

图4　查询处理器的结构

我们可以使用28个处理机执行这个并行数据流图，每个内结点由一个处理机执行，实现insert和join操作按流水线方式并行执行、join和两个scan操作按流水线方式并行执行、两个scan独立并行执行，也实现insert,join和两个scan本身的并行执行.

在一个CPDF图中，若存在一个从根到结点v的长度为k的路径，则称v为第k级结点.如果一个CPDF图的最高级结点的级数为n,则称这个CPDF图为n级CPDF图.n级CPDF图所有结点的级从0~$n-1$编号.

2　查询处理器

优化处理并行数据库查询的并行数据流方法需要一个查询处理器，其结构如图4所示.查询处理器由一个称为

© 中国科学院软件研究所 http://www.jos.org.cn

控制处理机的专门处理机执行.其它处理机称为查询处理机.

查询处理器的语法检查模块和顺序查询优化模块与传统关系数据库系统的相应程序模块相同.语法检查模块对用户查询进行语法检查、完整性验证、安全性检查,并将其变换为一棵以关系代数操作为结点的查询树.查询优化模块对查询树进行优化处理,产生一棵查询执行计划树.树的结点是指定了执行算法的关系代数操作.

查询处理器的生成SPDF图模块把顺序查询优化模块产生的查询执行计划树转换为SPDF图.这个转换比较容易,即对于查询计划树中的每条边(v,w),根据v和w的执行算法的特点,确定其是否可以按照流水线方式并行执行,为该边赋以权"p"或"s".

查询处理器的其余3部分比较复杂.生成CPDF图模块把输入的SPDF图变换为CPDF图.处理机分配模块为输入的CPDF图各结点分配处理机,生成并行查询执行计划.最后,并行查询计划调度执行模块协调多处理机执行并行查询执行计划,完成查询处理.

下边重点介绍CPDF图生成算法、处理机分配算法和并行查询计划调度执行算法.

3 CPDF图生成算法

给定一个SPDF图G,CPDF图生成算法分3步构造G的CPDF图:构造G的执行顺序图;划分G的结点;连接merge和split操作,形成G的CPDF图.下边介绍实现这3步所需要的算法.

3.1 执行顺序图的构造算法

给定一个SPDF图G,G的执行顺序图是一个单路径树,规定了数据操作的执行顺序.图中每个节点是G中可并行执行子树的集合,称为顺序执行分量.执行顺序图中各顺序执行分量由叶结点开始向上顺序执行.为了构造G的执行顺序图,我们先来构造G的数据相关图.

定义3.1. 给定SPDF图$G=(\langle V,E\rangle,W,F)$,$G$的数据相关图是加权树$T=\langle V',E'\rangle$,$V'$中每个结点是$G$的一个仅包含权为"$P$"的边的最大子图.对于任意$\alpha,\beta\in V'$,如果存在一条边$(v_1,v_2)\in E$,$v_1$属于$\alpha$,$v_2$属于$\beta$,则存在一条从$\alpha$到$\beta$的权为"$S$"的边$(\alpha,\beta)\in E'$.

下面是构造数据相关图算法.

输入:SPDF图$G=(\langle V,E\rangle,W,F)$

输出:G的数据相关图$G'=\langle V',E'\rangle$

(1) V':=空集合;E':=空集合;TMPG:=G;

(2) FORST:=删除TMPG中权为"S"的边所得到的森林;

(3) FOR FORST中每个子树t DO V':=$V'\cup\{t\}$;ENDFOR; * 为数据相关图建立一个新结点 *\

(4) FOR G中每条具有权"S"的边(v_1,v_2) DO

(5) E':=$E'\cup\{(t_1,t_2)\}$;(t_1,t_2)的权为"S"; * $t_1\in v'$,$t_2\in v'$,t_1包含结点v_1,t_2包含结点v_2 *\

(6) ENDFOR.

在构造执行顺序图之前,我们还需计算数据相关图中各结点的工作量.设数据相关图结点α包含SPDF图的结点$v_1,\ldots,v_k$.α的工作量定义为$\sum_{i=1}^{k}w_i$,其中w_i是v_i的工作量.计算数据相关图中各结点的工作量的关键是SPDF图各结点工作量的计算.

定义3.2. 设O是SPDF图的一个结点,ρ是输入关系集合.$W(O,\rho)=T_{cpu}+T_{i/o}$定义为$O$的工作量,其中$T_{cpu}$是在$\rho$上执行$O$所需要的处理机时间,$T_{i/o}$是在$\rho$上执行$O$所需要的磁盘读写时间.

$W(O,\rho)$可以根据O的实现算法和ρ中关系的大小,使用文献[12]的第11章和文献[13]的第7章提供的数学模型计算出来.

给定一个SPDF图G及其数据相关图T以及T中各结点的工作量,G的执行顺序图构造算法如下:

(1) 确定T的最长路径P;

(2) 建立一个N层框架(N是P的结点数),第i层框架是一个集合S_i;

(3) 对于$0\leqslant i\leqslant N-1$,把$P$的第$i$级结点分配到第$i$层框架$S_i$;

(4) 按照如下规则循环地分配T中P以外的结点:

a. 由叶到根顺序分配.

b. 第i级结点可以分配到满足条件$j\leqslant i$的任何框架S_j.

c. 只分配叶结点.

d. 当一个结点可分配到多个框架时,分配到具有最小工作量的框架.

e. 当一个结点被分配到一个框架后,从T中删除该结点及其所连的边.

(5) 建立执行顺序图:$S_{N-1}\rightarrow\ldots\rightarrow S_1\rightarrow S_0$.

© 中国科学院软件研究所 http://www.jos.org.cn

显然,执行顺序图的每个结点对应一个数据相关图的结点集合.由于数据相关图的每个结点对应一棵SPDF图的子树,而且这棵子树中的操作可以按照流水线方式并行执行,所以执行顺序图的每个结点对应一组SPDF图的可并行执行子树.显然,每个执行顺序图结点对应的操作集合可以并行执行,执行顺序图各结点必须按图规定的次序顺序执行.

3.2 结点划分算法

设O是SPDF图的一个结点.如果把O划分为k个可并行执行的兄弟结点,则O可以由k个处理机并行执行.由于SPDF图中具有多个结点,而且不同结点的工作量也不相同,所以在划分结点时必须综合考虑所有结点的工作量.工作量小的结点应该由较少处理机并行执行,即划分为较少兄弟结点.工作量大的结点应该由较多处理机并行执行,即划分为较多兄弟结点.

结点划分算法根据执行顺序图来划分SPDF图的结点.由于执行顺序图各结点必须顺序执行,所以每个执行顺序图结点对应的操作集合可以使用系统的所有处理机.结点划分算法以执行顺序图的结点所对应的SPDF图结点集合为单位,按执行顺序图结点的次序,划分SPDF图的每个结点.以下我们简称执行顺序图的结点为顺序执行结点,SPDF图的结点为操作结点.我们先给出计算操作结点划分比的方法.一个操作结点的划分比确定了它的并行度,即应分配给它的处理机个数.

定义 3.3. 设$OS(\alpha)=\{O_1,\ldots,O_k\}$是顺序执行结点$\alpha$对应的操作结点集合.对于$1\leqslant i\leqslant k$,$O_i$的划分比定义为$P_i=W(O_i,\rho)/(\Sigma_{O_j\in OS(\alpha)}W(O_j,\rho))$.

下面是计算结点划分比的算法.

输入:G:SPDF图;SEG:G的执行顺序图;W:存储G的各结点的工作量的数组.
输出:PR:存储G的各结点划分比的数组.
FOR SEG的每个顺序执行结点α DO
　　WORK$_\alpha$:=0;
　　FOR 每个数据操作结点$v\in$OS(α) DO WORK$_\alpha$:=WORK$_\alpha$+$W(v)$;ENDFOR;
　　FOR 每个结点$v\in$OS(α) DO PR(v)=$W(v)$/WORK$_\alpha$;ENDFOR;
ENDFOR.

下面是结点划分算法Partition的定义.给定一个SPDF图G及其执行顺序图SEG,Partition算法根据G中各结点的划分比,以SEG的结点所对应的操作结点集合为单位,把G的各结点划分为多个兄弟结点,扩展G图.

算法. Partition(G,SEG,PR,N)
输入:G:SPDF图;SEG:G的执行顺序图;PR:G中各结点的划分比;N:处理机数.
输出:EG:扩展的SPDF图.
FOR SEG的每个结点α DO
　　FOR α中每个操作结点v DO $P(v):=N\times PR(v)$(四舍五入);ENDFOR;
　　调整$\{P(v)\}$,使得$\sum_{v\in\alpha}P(v)=N$;
　　FOR α中每个G结点v DO $P(v)$个结点$v_1,\ldots,v_{P(v)}$加入EG;ENDFOR;
ENDFOR;
FOR G中每对父子结点(v,w) DO　　/* w已划分为$w_1,\ldots,w_{P(w)}$,v已划分为$v_1,\ldots,v_{P(v)}$ */
　　FOR i=1 TO $P(v)$ STEP 1 DO
　　　　FOR j=1 TO $P(w)$ STEP 1 DO
　　　　　　在EG中加入边(v_i,w_j);(v_i,w_j)的权为(v,w)的权;
　　　　ENDFOR;
　　ENDFOR;
ENDFOR.

3.3 merge和split操作的连接算法

算法Partition(G,SEG,PR,N)产生的扩展SPDF图不能直接执行,需要连接merge和split操作.下面我们来讨论merge/split操作的连接算法CSM.设G是输入到算法Partition的SPDF图,EG是算法Partition产生的并行数据流图.对G中每条边(β,α),EG中都有一个如图5所示的子图.其中$\alpha_1,\ldots,\alpha_k$是α的划分结果,$\beta_1,\ldots,\beta_m$是β的划分结果.若k和m皆为1,CSM算法对(β,α)不执行任何操作.否则,CSM算法根据定义1.3(2)的规则,在$\alpha_1,\ldots,\alpha_k$和$\beta_1,\ldots,\beta_m$之间连接merge和split操作结点.CSM算法如下.

算法. CSM(G,Partition,EG,DPR)
输入:G:简单并行数据流图;EG:G的扩展并行数据流图;DPR:Partition产生的结点划分结果.
输出:EEG:CPDF图.

© 中国科学院软件研究所 http://www.jos.org.cn

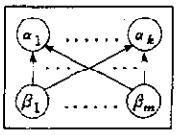

图5 EG中与G的边(β,α)对应的子图

(1) EEG_1=EG;
(2) FOR G 中每条边(β,α) DO
(3) Sa_1=EG 中由 α 划分而得的结点集$\{\alpha_1,\ldots,\alpha_k\}$;
(4) Sb_1=EG 中由 β 划分而得的结点集$\{\beta_1,\ldots,\beta_m\}$;
(5) IF $k=1\wedge m=1$ THEN GOTO 20; ENDIF;
(6) FOR $i=1$ TO k, $j=1$ TO m DO 删除 EEG 中边(β_j,α_i);ENDFOR;
(7) IF $m>1$
(8) THEN FOR Sa 中每个结点 α_i DO
(9) 在 EEG 中增加一个结点 merge(m);
(10) 在 EEC 中增加权为"P"的有向边(merge(m),α_i);
(11) ENDFOR;
(12) ENDIF;
(13) IF $k>1$ THEN
(14) FOR Sb 中每个结点 β_j DO
(15) 在 EEG 中增加一个结点 split(k);
(16) 在 EEG 中增加有向边$(\beta_j$,split(k)),权与(β,α)相同;
(17) 在 EEG 中增加权为"P"的有向边(split(k),MG_1),...,(split(k),MG_k); /* 如果 $m>1$,MG_i 是与 α_i 相连接的 merge(m)结点;如果 $m=1$,MG_i 是 α_i */
(18) ENDFOR;
(19) ELSE 在 EEG 中增加有向边$(\beta_1$,merge(m)),...,$(\beta_m$,merge(m)),权与(β,α)相同;
(20) ENDIF;
(21) ENDFOR.

4 处理机分配算法

设 Q 是一个查询,DF 是 Q 的 n 级 CPDF 图.下边是一个为 DF 分配处理机的简单算法.该算法假定系统具有充分多的处理机,可以为 DF 中每个结点分配一个处理机.

算法. ASSP(P,DF)
输入:处理机集合 $P=\{p_1,\ldots,p_n\}$,n 级 CPDF 图 DF.
输出:处理机分配方案.
(1) FOR $i=n-1$ TO 0 STEP -1 DO
(2) FOR DF 的每个 i 级结点 α DO
(3) IF α 是叶结点 THEN 为 α 分配一个处理机 p; $P:=P-\{p\}$;
(4) ELSE IF (α 与某个子结点 β 的连接边的权为"s")
(5) THEN 把分配给 β 的处理机分配给 α; /* 因为 α 必须在 β 之后执行 */
(6) ELSE 从 P 中选一个处理机 p 分配给 α; $P:=P-\{p\}$;
(7) ENDIF;
(8) ENDIF;
(9) ENDFOR;
(10) ENDFOR.

算法 ASSP 要求系统必须拥有相当多的处理机.下面我们修改算法 ASSP 以适应一般情况.

定义 4.1. 设 DF=$(\langle V,E\rangle,W,F)$是一个 CPDF 图.DF 的子 CPDF 图是一个 CPDF 图$(\langle V_1,E_1\rangle,W,F_1)$,其中,$\langle V_1,E_1\rangle$是$\langle V,E\rangle$的子树,$F_1$ 除了定义域为 E_1 以外与 F 相同.

定义 4.2. 设 DF=$(\langle V,E\rangle,W,F)$是一个 CPDF 图.DF 的一个根为 r 的子 CPDF 图 SDF 称为 DF 的一个完全子 CPDF 图,如果 SDF 中包含了 r 的所有后裔结点及其相关的加权边.

定义 4.3. 设 DF=$(\langle V,E\rangle,W,F)$是一个 CPDF 图.DF 的完全子 CPDF 图 SDF_{max}称为 DF 的满足性质 P 的最大完全子 CPDF 图,如果 DF 中不存在满足性质 P 的完全子 CPDF 图 SDF,使得 SDF_{max}是 SDF 的子图.

我们先给出一个 CPDF 图分解算法 DEC.给定一个 CPDF 图 DF 和一个正整数 K,DEC 算法构造一个与 DF 等价的 CPDF 图森林 FOREST,其中每个 CPDF 图都是 DF 的完全子 CPDF 图,可同时执行的结点数不超过 K.森林中的 CPDF 图是有序的.这里的等价是指 DF 与 FOREST 的执行结果相同.DEC 算法的定义如下.

算法:DEC(DF,K)
输入:n 级 CPDF 图 DF,正整数 K.
输出:CPDF 图森林 FOREST,每个 CPDF 图中可并行执行的结点数不超过 K.
(1) $i_1:=0$;

© 中国科学院软件研究所 http://www.jos.org.cn

(2) WHILE DF≠空 DO
(3)　　求 DF 的最大完全于 CPDF 图 T,使 T 中可并行执行的结点数≤K;
(4)　　把 T 之根结点在 DF 中的父结点与 T 对应的操作量改为临时关系 R_T;　　/* T 的结果存入 R_T */
(5)　　T 添到 FOREST 最后,作为第 i 个 CPDF 图 DF_i;
(6)　　在 DF 中删除 T;
(7)　　$i:=i+1$;
(8) ENDWHILE.

使用算法 DEC 和 ASSP,我们可以定义一个一般化的处理机分配算法:

算法. IASSP(P,DF)

输入:具有 K 个处理机标识符的集合 P,n 级简单并行数据流图 DF.
输出:处理机分配方案.
(1) 执行算法 DEC(DF,K),建立森林 FOREST;
(2) FOR FOREST 中的每个 CPDF 图 T DO
(3)　　执行算法 ASSP(P,T);
(4) ENDFOR.

5 并行查询执行计划调度执行算法

现在我们来讨论并行查询计划(即 CPDF 图)的调度执行算法(以下简称查询执行算法).查询执行算法按照 CPDF 图调度多个处理机,并行执行用户查询.下面是与 ASSP 算法相对应的查询执行算法.

算法. QE(PQP)

输入:PQP:并行查询执行计划,即 n 级 CPDF 图.
输出:查询结果.
(1) WHILE (PQP 中存在未启动执行的结点) DO
(2)　FOR $i=n-1$ TO 0 STEP -1 DO
(3)　　FOR PQP 中第 i 级未启动执行的结点 α DO
(4)　　IF (α 是叶结点)
(5)　　THEN 启动分配给 α 的处理机,执行 α 所对应的操作;
(6)　　ELSE IF (α 与其所有于结点的连接边皆有权"s")
(7)　　THEN IF (α 的所有后裔结点执行结束)
(8)　　THEN 启动分配给 α 的处理机,执行 α 所对应的操作;
(9)　　ENDIF;
(10)　　ELSE 启动分配给 α 的处理机,执行 α 所对应的操作;
(11)　　ENDIF;
(12)　　ENDIF;
(13)　　ENDFOR;
(14)　ENDFOR;
(15) ENDWHILE.

算法 QE 在整个并行查询计划所有结点对应的处理机被启动之前,控制处理机始终执行 WHILE 循环.在下边两种情况下,QE 将浪费控制处理机的大量时间.

(1) 并行查询执行计划具有较多的权为"s"的边;
(2) 存在权为"s"的边(A,B),而且以 A 为根的子树需要很长的时间才能结束执行.

为了改进 QE 算法,我们首先去掉 QE 算法中的 WHILE 循环语句,使之成为如下过程:

QE′(PQP)
(1) IF (PQP 中有未启动执行的结点)
(2) THEN 同 QE 算法的(2)~(14)步;
(3) ELSE 并行查询执行计划执行完毕;
(4) ENDIF.

使用过程 QE′,并行查询执行计划的执行过程如下:
(1) 控制处理机初始执行一次算法 QE′(PQP);
(2) 控制处理机利用空闲时间执行其它程序;
(3) 当任意一个执行查询操作的处理机结束时,通知控制处理机,再一次执行算法 QE′(PQP);
(4) 重复步骤(2)和(3),直至并行查询执行计划执行完毕.

并行查询执行计划的这种执行过程可以使控制处理机利用空闲时间完成其它任务,提高了控制处理机的利用率.

© 中国科学院软件研究所 http://www.jos.org.cn

相应于算法 IASSP 的查询计划调度执行算法如下：

算法. IQE(FOREST).

输入：FOREST=$\{T_1,T_2,\ldots,T_m\}$:CPDF 图森林.

输出：查询结果.

(1) FOR i=1 TO m STEP 1 DO

(2) 执行算法 QE(T_i)； /* 此步骤也可以由过程 QE'实现 */

(3) ENDFOR.

参考文献

1 李建中.并行数据库的查询处理并行化技术和物理设计方法.软件学报,1994,5(10):1～10
(Li Jian-zhong. Parallelization techniques for query processing and data declustering methods. Journal of Software, 1994,5(10):1～10)

2 李建中.并行数据操作算法和查询优化技术.软件学报,1994,5(10):11～23
(Li Jian-zhong. Parallel data operation algorithms and query optimization techniques. Journal of Software, 1994,5(10):11～23)

3 Deen S M, Kannangara D N P, Taylor M C. Multi-join on parallel processors. In: Deen S M ed. Proceedings of the 2nd International Symposium on Databases in Parallel and Distributed Systems. Los Alamitos, California, USA: IEEE CS Press, 1990. 92～102

4 Stonebraker M *et al*. The design of XPRS. In: Bancilhon F, DeWitt D J eds. Proceedings of the 18th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1988. 318～330

5 Lu L, Shan M C, Tan K L. Optimization of multi-way join queries for parallel execution. In: Lohman G M, Sernadas A, Camps R eds. Proceedings of the 17th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1991. 549～560

6 Chen M S *et al*. Using segmented right-deep trees for the execution of pipelined hash joins. In: Yuan L Y ed. Proceedings of the 18th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1992. 15～26

7 Lanzelotte R S G, Valduriez P, Zait M. On the effectiveness of optimization search strategies for parallel execution spaces. In: Agrawal R, Baker S, Bell D eds. Proceedings of the 19th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1993. 493～504

8 Schneider D A, Dewitt D J. Tradeoffs in processing complex join queries via hashing in multiprocessor database machines. In: Mcleod D, Sacks-Davis R Schek H eds. Proceedings of the 16th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1990. 469～479

9 Shekita E J, Young H C, Tan K L. Multi-join query optimization for symmetric multi-processors. In: Agrawal R, Baker S, Bell D eds. Proceedings of the 19th International Conference on Very Large Data Bases. San Mateo, California: Morgan Kaufman, 1993. 479～492

10 DeWitt D J, Gray J. Parallel database systems: the future of high performance database systems. Communications of the ACM, June 1992,35(6):85～98

11 Graefe G. Encapsulation of parallelism in the Volcano query processing system. In: Garcia-Molina H ed. Proceedings of ACM SIGMOD'90. Baltimore, USA: ACM Press, 1990. 102～111

12 Ullman J D. Principles of database and knowledge base systems, Vol Ⅱ. Rock Ville, Maryland, USA: Computer Science Press, Inc., 1989

13 李建中,孙文隽.并行关系数据库管理系统引论.北京:科学出版社,1997
(Li Jian-zhong, Sun Wen-jun. Introduction to parallel relational database systems. Beijing: Science Press, 1997)

Parallel Dataflow Method for Optimizing and Processing Queries on Parallel Databases

LI Jian-zhong

(*Information Research Institute Heilongjiang University Harbin* 150080)

Abstract The focus of this paper is on how to use the parallel dataflow technique to optimize and process queries on parallel databases. A set of algorithms for implementing the query optimizers of parallel database systems based on the parallel dataflow technique, such as the algorithms for generating parallel dataflow graph from a query, the algorithms for processor assignment, and the algorithms for scheduling and executing the parallel query plan, are proposed. A parallel query optimizer using the algorithms are presented also in the paper. The algorithms and the optimizer have been used in a prototype parallel database system designed by the author. It is shown that the proposed algorithms and the optimizer are very efficient and effective for optimizing and processing queries on parallel databases.

Key words Parallel databases, query optimization, parallel dataflow, parallel query optimization.

Class number TP311.13

© 中国科学院软件研究所 http://www.jos.org.cn