并行数据库的查询处理并行化技术和物理设计方法*

李建中

(黑龙江大学信息技术研究所,哈尔滨 150080)

摘要 近几年,随着并行计算机系统的迅速发展,并行数据库系统已经成为一个新的数据库研究领域,引起了学术界和工业界的极大关注,很多研究成果已经出现.本文是综述并行数据库系统研究与进展情况的两篇文章之一,重点探讨目前并行数据库系统的研究方向和问题,综述有关并行数据库的物理设计方法和查询处理并行化技术的主要研究成果.

关键词 并行数据库,数据划分方法,并行数据流方法.

近 10 年来,并行计算机系统的发展十分迅速,很多商品化的高性能并行计算机系统已经出现.但是,并行计算机软件系统的发展却远远落后于硬件的发展,并行软件系统的研究与开发已经成为急需解决的问题.由于在各种应用领域中数据库系统占有重要的地位,并行数据库系统的研究引起了学术界和工业界的特别关注.特别是,随着计算机应用领域的不断扩大,数据库的规模越来越大,数据库的查询也越来越复杂,传统的数据库系统已经难以适应迅速增长的应用要求.人们正在迫切期望高性能数据库系统的到来.并行计算机系统的出现为高性能数据库系统的实现带来了希望.并行数据库系统必将成为未来的高性能数据库系统.

并行数据库系统的研究可以追溯到数据库机器的研究[1].数据库机器的研究致力于实现数据操作并行化的专用硬件的设计.由于磁盘和处理机之间速度差别的迅速增长和数据库机器硬件设计的高开销,数据库机器的研究失败了.但是,随着通用并行计算机系统的出现,并行数据库系统的研究取得了很大的进展.几个成功的并行数据库系统原型都建立在通用并行计算机系统上.并行数据库系统已经成为并行计算机的主要应用之一.近几年,以通用并行计算机为基础的并行数据库系统的研究出现了高潮[1,2].很多研究者认为,在具有数百个甚至上千个处理机的并行计算机上建立并行数据库系统是可行的.目前,并行数据库系统的研究基本围绕着关系数据库进行,主要包括以下三个方面:1. 并行数据库的物理设计方法;2. 并行数据操作算法的设计;3. 并行数据库系统的查询优化.

* 本文 1993-07-11 定稿

本项目是国家自然科学基金资助项目,并受到国家教委优秀青年教师基金、黑龙江省科委和哈尔滨市科委的资助.

作者**李建中**,44 岁,教授,主要研究领域为并行软件系统与并行数据库系统.

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

本文将首先讨论各种支持并行数据库系统的并行计算机结构和关系数据库查询的固有并行性,然后分别介绍查询处理的并行化技术和并行数据库的物理设计方法.在本文的续篇中,我们将综述并行数据操作算法和并行数据库的查询优化技术,并探讨今后并行数据库系统的研究方向和问题.

1 支持并行数据库系统的并行结构

目前,并行数据库系统研究所依赖的并行计算机结构主要包括四种:完全共享结构(SHARED EVERYTHING,以下简称 SE 结构)、共享主存储器结构(SHARED MEMORY,以下简称 SM 结构)、无共享资源结构(SHARED NOTHING,以下简称 SN 结构)、共享磁盘结构(SHARED DISK,以下简称 SD 结构).在 SE 结构中,处理机共享主存储器和磁盘存储器,处理机、共享主存储器和磁盘存储器由高速通讯网络连接.处理机之间的通讯可以通过主存间接实现,也可以通过高速通讯网络直接进行.具有 SE 结构的并行计算机包括 SEQUENT SYMMETRY、FIREFLY、SPUR 等系统.在 SM 结构中,多处理机共享主存储器,每个处理机具有独立的磁盘存储器,多处理机和共享主存由高速通讯网络连接.处理机之间的通讯可以通过主存间接实现,也可以通过高速通讯网络直接实现.IBM/370 多处理机系统、VAX 多处理机系统和 SEQUENT 系统是具有 SM 结构的典型并行计算机系统.在 SN 结构中,没有任何共享硬件资源,每个处理机都具有独立的主存储器和磁盘存储器,处理机之间的通讯由高速通讯网络实现.TERADATA 系统、TANDEM 系统、NCUB 系统和较新的 VAXcluster 系统都具有 SN 结构.在 SD 结构中,处理机共享所有磁盘存储器,每个处理机具有独立的主存储器,处理机之间的通讯可以通过磁盘存储器间接实现,也可以通过高速通讯网络直接实现.IBM 的 SYSPLEX 系统就是一个基于 SD 结构的并行计算机系统.

1986 年,Stonebraker 提出 SN 结构是支持并行数据库系统的最好并行结构[3].目前,这个观点已经被普遍的接受.SN 结构的优点主要有三个.首先,SN 结构通过最小化共享资源使得由资源竞争带来的系统干扰最小化;其次,SN 结构具有高可扩充性,处理机个数可扩展到数百甚至上千个而不增加处理机间的干扰;第三,SN 结构在复杂数据库查询处理和联机事务处理过程中可获得接近线性的加速.本文主要基于 SN 结构.以下,我们称 SN 结构中每个由单处理机、主存储器和磁盘构成的单处理机系统为处理结点.

2 关系数据库查询的固有并行性

关系模型和关系数据库系统的非过程性查询语言为其并行实现提供了有利的条件.关系查询特别适于并行处理.关系查询一般由多个基本数据操作组成.这些数据操作与关系集合形成了一个代数系统,称为关系代数.由于关系代数的封闭性和数据操作的相对独立性,关系查询具有三种固有并行性,即数据操作间的流水线并行性(Pipelining parallelism)、数据操作间的独立并行性(Inter—operator parallelism)和单数据操作内的并行性(Intra—operator parallelism)[4].

设$(6,Я)$是关系代数系统,其中 6 是数据操作集合,Я 是关系集合.关系查询是一个$(6,Я)$上的代数表达式.以下,Q 表示关系查询,$OP\in 6$ 表示 Q 中的关系操作,$INPUT(OP)$

$\in$ℜ表示OP的输入关系,$OUTPUT(OP)\in$ℜ表示OP的输出关系.

定义 2.1. 设$B=(r,w,S)$是一个主存缓冲区,其中,S是用于存储数据的队列,w是写指针,r是读指针,w和r的初始值皆为空.

(1)操作$PRODUCE(X,B,k)$的定义如下:

IF (S中可用空间$<k$) THEN 等待S可用空间$\geqslant k$;

X中k个数据传送到S的第w到第$w+k-1$单元;

调整指针w.

(2)操作$CONSUME(X,B,k)$的定义如下:

IF (S中数据个数$<k$) THEN 等待S中数据个数$\geqslant k$;

S的第r到第$r+k-1$单元的数据传送到X;

调整指针r.

定义 2.2. 设OP_1、$OP_2\in$6. 如果$OUTPUT(OP_1)\cap INPUT(OP_2)\neq\theta$,则称$OP_2$直接依赖于$OP_1$. 如果存在$OP_1$、…、$OP_k\in$6使得

$$OUTPUT(OP_1)\cap INPUT(OP_2)\neq\theta,\cdots,OUTPUT(OP_{k-1})\cap INPUT(OP_k)\neq\theta,$$

则称OP_k依赖于OP_1. 如果两个数据操作无依赖关系,则称这两个操作是相互独立的.

定义 2.3. 设OP_1、$OP_2\in$6. 如果OP_1和OP_2相互独立,则称OP_1和OP_2是可独立并行执行的(Inter-operator parallellism).

定义2.3可以扩展成m(>2)个数据操作可独立并行执行的定义. 在一个关系查询中常常存在多个可独立并行执行的数据操作. 我们可以使用多个处理机同时执行这些可独立并行执行的操作,实现查询处理的并行化.

定义 2.4. 设OP_1、$OP_2\in$6,$B=(w,r,S)$是一个主存缓冲区. 如果OP_1中至少包含一个$PRODUCE(OUTPUT(OP_1),B,k_1)$操作,$OP_2$中至少包含一个$CONSUME(INPUT(OP_2),B,k_2)$操作,则称$OP_2$在缓冲区$B$上以流水线方式依赖于$OP_1$.

定义 2.5. 设OP_1、$OP_2\in$6. 如果OP_2(或OP_1)在缓冲区B上以流水线方式依赖于OP_1(或OP_2),则称OP_1和OP_2可以按流水线方式并行执行(Pipelining parallellism).

两个数据操作可以按流水线方式并行执行是指一个操作(producer)的输出是另一个操作(consumer)的输入. 我们可以为每个操作分配一个处理机,使得producer产生第一个输出后consumer即可执行,从而实现了这两个操作的并行执行. 请注意,在一个关系查询中,可按流水线方式并行执行的数据操作的个数不局限于两个,可以有任意多个.

定义 2.6. 设$OP\in$6. 若OP可以分为多个可独立或按流水线方式并行执行的子任务,则称OP是可并行执行的操作(Intra-operator parallellism).

上述定义2.3、2.5和2.6定义了关系数据库查询的三种固有并行性. 这三种并行性为成功地建立并行关系数据库系统提供了充分的条件.

3　查询处理的并行化技术

本节主要介绍实现查询处理并行化的并行数据流方法. 并行数据流方法的核心是使用现有的数据操作算法并行地执行关系数据库查询,实现关系数据库查询的三种固有并行性.

并行数据流方法既不需要设计新的并行数据操作算法,也不需要修改现有的顺序数据操作算法,是一种在现有数据库理论、技术和方法的基础上实现并行数据库系统的简单方法.并行数据流方法分为简单并行数据流方法和以数据划分为基础的并行数据流方法.这些方法已被GAMMA、VOLCANO、TANDEM等系统使用[5-7].

我们首先讨论简单并行数据流方法.关系查询可以作为并行数据流图执行.图1给出了一个SQL查询语句及其并行数据流图.图中scan表示数据操作选择和投影的组合(select—project).如果为图中每个操作分配一个处理机,并按照数据流图执行给定的查询,则可以使insert和join按流水线方式并行执行,join和两个scan按流水线方式并行执行,两个scan独立并行执行,实现了该查询的并行执行.使用简单并行数据流图方法实现关系查询的并行化,需要一个关系查询执行器,完成以下几个功能:1.由查询语句产生数据流图;2.为数据流图中的数据操作分配处理机;3.协调各个处理机调用顺序数据操作算法完成查询处理.

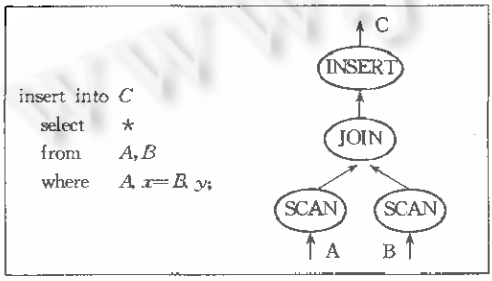

图1 一个SQL查询语句及其并行数据流图

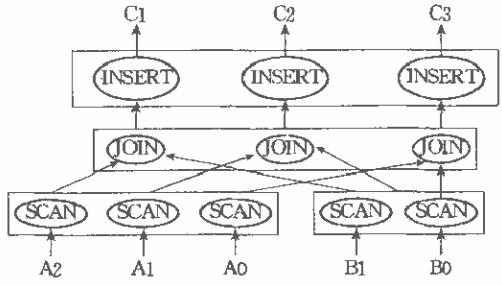

图2 一个基于数据划分的并行数据流图

简单并行数据流图方法可以有效地发挥关系数据库查询的流水线并行性和独立并行性,但是难以实现单关系操作的并行性.以数据划分为基础的并行数据流图方法(简称PPDA方法)可以很好地解决这个问题.

PPDA方法假设系统中的关系都已根据需要划分为多个子集合,存储在不同的磁盘上.当系统接收一个查询后,PPDA的查询运行器将其转化为并行数据流图,然后按照这个并行数据流图进行查询处理.PPDA处理查询中每个数据操作OP的方法与简单并行数据流图方法不同.它首先把操作关系划分为N个子集合,并建立N个执行相同操作OP的进程,每个进程在一个子集合上运行;然后,使用多个处理机并行地执行这些进程;最后,将执行结果合成为OP操作的最终结果.

在简单并行数据流图方法中,每个一元数据操作具有一个输入数据流和一个输出数据流;每个二元数据操作具有两个输入数据流和一个输出数据流.它们与顺序数据操作算法的要求是一致的.所以,简单并行数据流图方法可以直接使用现有的数据操作算法实现查询的并行处理.但是,在PPDA中,由于数据的划分和数据操作的多进程化,每个数据操作进程可能具有多个输入数据流,而且每个操作进程的输出数据流可能需要分解为多个数据流,为其它操作进程提供输入数据.于是,除了查询运行器以外,PPDA方法还需要两个新的数据操作,即合并(merge)和分解(split)操作.合并操作的功能是将多个输入数据流合并为一个输出数据流.分解操作的功能是将一个输入数据流分解为多个输出数据流.合并和分解操作

的算法不是唯一的,与数据划分策略有关.合并和分解操作算法一般都具有缓冲处理功能.把合并和分解操作与数据操作进程相结合,我们就可以使用 PPDA 方法利用现有的顺序数据操作算法实现查询的并行处理.

下面,我们以图 1 中SQL 语句为例,说明如何用 PPDA 方法实现查询的并行处理.设关系 A 已分解为三个子集合 A_0、A_1 和 A_2,存放在三个不同的磁盘上;关系 B 已分解为两个子集合 B_0 和 B_1,存放在两个不同的磁盘上;关系 C 已分解为三个子集合 C_0、C_1 和 C_2,存放在三个不同的磁盘上.根据 A、B 和 C 的划分情况,查询运行器为 A 上的 $SCAN$ 操作建立三个 $SCAN$ 进程,记作 $SCAN_1$、$SCAN_2$ 和 $SCAN_3$;为 B 上的 $SCAN$ 操作建立两个 $SCAN$ 进程,记作 $SCAN_4$ 和 $SCAN_5$;为 $JOIN$ 操作建立三个 $JOIN$ 进程,记作 $JOIN_1$、$JOIN_2$ 和 $JOIN_3$;为 $INSERT$ 操作建立三个 $INSERT$ 进程,记作 $INSERT_1$、$INSERT_2$ 和 $INSERT_3$.给定查询的基于数据划分的并行数据流图如图 2 所示.

为了保证数据流正确地在各个进程之间流动,我们需要在每个 $SCAN$ 进程的输出端连接一个分解操作,在每个 $JOIN$ 进程的输入端连接一个合并操作,在每个 $JOIN$ 进程的输出端连接一个分解操作,在每个 $INSERT$ 进程的输入端连接一个合并操作.关系 A 的 $SCAN$ 进程输出端所连的分解操作的功能是将 $SCAN$ 的输出按 $JOIN$ 操作的数据划分要求分解为三个数据流,分别送到三个 $JOIN$ 进程的第一输入端所连的合并操作.关系 B 的 $SCAN$ 进程输出端所连的分解操作的功能是将 $SCAN$ 的输出按 $JOIN$ 操作的数据划分要求分解为三个数据流,分别送到三个 $JOIN$ 进程的第二输入端所连的合并操作.$JOIN$ 进程输出端所连的分解操作的功能是将 $JOIN$ 的输出按关系 C 的划分策略分解为三个数据流,分别送到三个 $INSERT$ 进程输入端所连的合并操作.所有合并操作的功能相同,都是把多个输入数据流合并为一个数据流,传输给所连的数据操作进程.如果查询运行器为每个数据操作进程和相应的分解与合并操作分配一个处理结点,按照图 2 所示的并行数据流图执行给定的数据库查询,则可实现操作 $INSERT$ 和 $JOIN$、$JOIN$ 和两个 $SCAN$ 之间的流水线并行运行以及两个 $SCAN$ 间的独立并行运行,也可实现 $INSERT$、$JOIN$ 和两个 $SCAN$ 本身的并行运行.

4　并行数据库的物理设计方法

并行数据库物理设计的核心问题是:如何把一个关系划分为多个子集合并分布到多个处理结点上(以下简称数据划分),使得在查询处理中系统的并行性得到充分发挥.在第 3 节我们已经看到,数据划分是查询处理并行化的重要基础.很多研究表明,数据划分对于并行数据库系统的性能具有非常大的影响.数据划分是并行数据库系统的一个重要研究领域,已经出现了很多数据划分方法.这些方法可以分为三类:一维数据划分、多维数据划分和传统物理存储结构的并行化.

4.1　一维数据划分

一维数据划分是最简单的数据划分方法,已经被 Arbre、Bubba、Gamma、Teradata 等系统采用[5,7,8].一维数据划分方法的特点是根据关系的一个属性的值来划分整个关系.这个属性称为划分属性.目前已经提出了四种一维数据划分方法.以下设并行数据库系统具有

N 个处理结点,编号为 0、…、$N-1$, $R(A_1,\cdots,A_k)$是具有属性 A_1、…、A_k 的关系.

4.1.1 Round－Robin 划分方法

Round－Robin 方法并不是一维数据划分方法.因为它和一维数据划分方法一样简单,所以我们也把它归并到这一类.设 r_i 是关系 R 的第 i 个元组.使用 Round－Robin 划分方法,r_i 将被存储到第(i mod N)个处理结点上.对于大数据量查询,Round－Robin 方法是很理想的.但是,它不能有效地支持具有低选择性谓词的查询.这是因为这种查询仅存取很少的元组,可 Round－Robin 方法却要求启动所有的处理结点,增加了系统开销.

4.1.2 Hash 划分方法

设 A 是关系 R 的划分属性,V 是 A 的值域.使用 Hash 划分方法,需要定义一个以 V 为定义域的函数 $H:V\rightarrow\{0,\cdots,N-1\}$.设 r 是 R 的任意元组.Hash 划分方法把 r 存储到处理结点 $H(r[A])$上,其中 $r[A]$表示元组 r 在属性 A 上的值,Hash 方法既能有效地支持大数据量存取操作,也能有效地支持在划分属性上具有低选择性谓词的数据操作.然而,Hash 方法不能有效地支持具有小区域存取要求的查询操作,也容易引起数据在处理结点间分布不均匀.

4.1.3 Range 划分方法

Range 划分方法把划分属性 A 的值域划分为 N 个区间 $I_0=[x_0,x_1)$、…、$I_{N-1}=[x_{N-1},x_N)$.然后,将 R 划分为 N 个子集合 S_1、…、S_2,其中,$S_i=\{r|r\in R,r[A]\in I_i\}$.$S_i$ 存储在第 i 个处理结点上.Range 划分方法既可以有效地支持大数据量存取操作和在划分属性上具有低选择性谓词的数据操作,也完全支持基于划分属性的区域查询操作.但是,Range 划分方法可能引起数据在处理结点间分布的不均匀,在最坏情况下,关系中的所有数据可能皆分布在同一个处理结点上.

4.1.4 Hybrid－Range 划分方法[9](简称 HR 方法)

对需要消耗大量系统资源(CPU 时间、I/O 时间、通讯时间)的查询来说,所涉及的关系划分越细,分布的处理结点数越多,参与处理该查询的处理结点就越多,查询处理的效率也就越高.但是,对于消耗较少系统资源的查询来说,加细关系划分粒度可能会增加查询的响应时间.这是因为多处理结点的启动、通讯和终止的额外开销可能会大于并行处理所带来的时间节省量.总之,关系划分策略的选择对于系统的整体性能有重要的影响.上述三种划分方法没有考虑这些影响.HR 划分方法试图在考虑这些影响的条件下给出关系的优化划分策略.它首先确定子集合的大小 FC;然后,按划分属性值对 R 进行排序;次之,把 R 划分为 |R|/FC 个子集合;最后,按 Rand－Robin 方式在处理结点间分布各子集合.HR 方法的关键是 FC 的确定.FC 是在最小化关系 R 上平均查询响应时间的条件下被确定的.HR 方法试图使要求较少系统资源的查询由较少处理结点处理,而使要求较多系统资源的查询由较多处理结点处理,提高了系统的查询效率和处理能力.虽然 HR 方法克服了其它一维数据划分方法的缺点,由于它根据数据操作资源需求量的均值确定 FC,当数据操作资源需求量的方差很大时,它将失效.HR 方法是基于选择操作建立的.它是否能有效地支持其它数据操作尚需研究.

4.2 **多维数据划分**

一维数据划分方法具有一个共同的问题:不能有效地支持在非划分属性上具有选择谓

词的查询. 这样的查询必须被送到所有包含给定关系元组的处理结点进行处理. 显然，把查询送到不包含所涉及元组的处理结点运行将浪费大量的 CPU 时间、I/O 时间和通讯时间，降低系统的处理能力. 最近，为了解决这个问题，几个多维数据划分方法已经被提出.

4.2.1　CMD 多维数据划分方法

本文作者提出了第一种多维数据划分方法，称为 CMD 方法[10]. CMD 方法视具有 d 个属性的关系 R 为 d 维空间 S 上的子集合. 它首先根据 R 中数据的分布把空间 S 的各维划分为多个不相交区间，把空间 S 划分为多个超方体，使得 R 在每个超方体中具有近似相同的元组数. 然后，使用坐标和求模函数把每个超方体唯一地分配到一个处理结点. 如果 d 维超方体 H 具有坐标 $(X_1,\cdots,X_d)$，则 H 被分配到处理结点 $\mathrm{CMD}(X_1+\cdots+X_d)\ mod\ N$，其中，$N$ 是处理结点个数. S 在各处理结点上的子空间是极不规则的，使得 R 在各结点上的子集合难以使用各种现存的物理存储结构组织. 为了解决这个问题，CMD 方法通过坐标变换，把每个处理结点上的子空间压缩成为一个超方体，使得 R 在该子空间上的子集合在变换后的子空间上近似均匀分布. CMD 方法使用 GRID 文件结构在各处理结点上组织 R 的子集合. 理论和实验结果都表明，CMD 方法在多数情况下都是优化的，即能够充分发挥系统的并行性. 由于数据划分对称地在所有属性上进行，CMD 方法可以有效地支持具有各种选择谓词的查询. 经过 CMD 方法划分的关系是部分排序的. 该性质使基于 CMD 的 SORT 和 JOIN 等数据操作的实现算法远比现有的算法有效[11,12]. CMD 方法把由数据维护操作引起的超方体的分裂与合并局部于单个处理结点，减少了处理结点间的数据传输，提高了数据维护操作的效率. 对于所有可预知其数据分布的关系，CMD 方法都是均衡的，即每个处理结点上的数据子集合的大小都近似相等，避免了为保持数据均衡而进行的耗时的数据再划分操作.

4.2.2　BERD 多维数据划分方法[13]

BERD 方法通过划分多个属性实现关系的划分. 它把多个划分属性分为两类. 一类称为主划分属性，一个关系只有一个. 另一类称为辅助划分属性，一个关系可以有多个. 给定关系 R，令 A 是主划分属性，B_1、…、B_k 是辅助划分属性. BERD 首先以 A 为划分属性，使用 Range 划分方法划分 R. 然后，对于每个辅助属性 B_i，BERD 建立一个辅助关系 $RB_i[V_i,K_i,P_i]$，其中，V_i 与 B_i 的值域相同，T_i 与 R 的关键字的值域相同，P_i 的值域是所有处理结点号的集合. 元组 $(v,k,p)\in RB_i$ 当且仅当存在一个元组 $T\in R, T[B_i]=v, T$ 的关键字值是 k，p 是 T 所在的处理结点号. 每个辅助关系使用 RANGE 划分方法分布到多个处理结点. 对于在主划分属性 A 上具有选择谓词的查询，查询运行器将根据 R 按属性 A 划分的情况，令所有包含有关元组的处理结点并行地处理这个查询. 如果用户提交一个在辅助划分属性 B 上具有选择谓词的查询，查询运行器首先利用辅助关系 RB 确定哪些处理结点包含有关的元组，然后，令这些处理结点并行地处理这个查询. BERD 方法具有类似于 Range 划分方法的问题. 此外，它在处理辅助划分属性上具有区间谓词的查询时，需要搜索辅助关系，系统开销很大.

4.2.3　MAGIC 多维数据划分方法

DeWitt 等提出了第三种多维数据划分方法，称为 MAGIC 方法[13]. MAGIC 方法是 Hybrid－Range方法的推广. 为了使用 MAGIC 方法划分关系 R，用户需要说明 R 的 k 个划分属性、R 上所有选择操作的系统资源需求量和它们运行的频率. 类似于 Hybrid－Range

方法,MAGIC 根据用户给定的信息确定划分后 R 的子集合大小 FC 和 k 个划分属性值域的划分策略,最小化所有选择操作的平均响应时间. MAGIC 方法具有类似于 Hybrid－Range 方法所具有的问题.

4.3 传统物理存储结构的并行化

实践证明,很多现有的单处理机系统上的文件结构都是非常有效的数据库物理存储结构.把这些文件结构应用于并行数据库系统显然是有益的.目前,这方面的工作还不多,仅存在几种 B－树和 GRID 文件的并行化方法.

4.3.1 B－树的并行化方法

从文献[14]讨论的三种多磁盘 B－树方法,我们可以得到三种 B－树的并行化方法.第一种方法称为记录划分方法.这种方法首先在每个处理结点上建立一个 B－树,然后使用 RANGE 划分方法、HASH 划分方法或 Round－Robin 方法把初始 B－树的记录分配并插入到各个处理结点的 B－树中去.第二种方法称为大页 B－树方法.在通常的 B－树中,一个结点只占一个物理页.大页 B－树方法则令每个 B－树结点占多个物理存储页,称为超页.每个超页被划分为 N 个子集合,并被分配到不同的处理结点上.第三种方法称为页划分方法.这种方法在 N 个处理结点间分布通常 B－树的各结点页.B－树的每个索引项需要增加一个信息,说明该索引项所对应的页所在的处理结点.页的分布可按 Rand－Robin 等方法进行.为了保证局部工作负载平衡,页划分方法可加以改进:如果在 B－树的某一级上插入一页,这页将被分配到与以此页为中心的 $N-1$ 或 $N-2$ 个相邻页所在的处理结点不同的处理结点.这样,在 B－树的任一级上,任意小于等于 $N/2$ 个相邻页都存储在不同的处理结点上.如果在 B－树的某一级把一页与其邻页合并,为了保证任意小于等于 $N/2$ 个相邻页都存储在不同的处理结点上,需要动态调整页的分布.

4.3.2 第一种 GRID 文件并行化方法

Hua 和 Lee 提出了第一种 GRID 文件的并行化方法[15].这种方法通过在多处理结点间分布 GRID 块实现 GRID 文件的并行化.它需要对多维目录做一个小的修改,使得每个目录项包含所对应的 GRID 块中的记录数及其所在的处理结点,而不再包含其所在的磁盘页号.这种方法需要解决的关键问题是 GRID 文件在多处理结点间的初始分布和由数据插入与删除所引起的数据负载不平衡的处理.文献[15]给出的初始数据分布算法首先按 GRID 块中记录数的递减顺序把所有 GRID 块排序,建立一个 GRID 块有序表.然后,按照排序后的顺序把 GRID 块分配到各处理结点.分配的方法如下:把 GRID 块有序表中最大(即具有最多记录)的 GRID 块分配到最小(即具有最少记录)的处理结点上;从 GRID 块有序表中删除已分配的 GRID 块;重复此过程直至 GRID 块有序表空为止.为了减少由插入与删除所引起的处理结点间的数据不平衡,文献[15]给出了一个数据再分布算法.这个算法的目标是以最小的开销减少处理结点间的数据负载不平衡.文献[15]给出的并行化方法有两个弱点.一是当处理结点间的数据负载很不平衡时,上述数据再分布算法可能失效.二是很多相邻 GRID 块可能被分配到相同的处理结点上,可能引起工作负载的不平衡.

4.3.3 第二种 GRID 文件并行化方法

本文作者提出了第二种 GRID 文件并行化方法[16].给定一个文件 F,这种方法首先构造一个逻辑 GRID 文件,其 GRID 块可以任意大,所有 GRID 块的大小近似相同.然后,它使用

CMD方法在多处理结点间分布这个逻辑GRID文件，不但使处理结点间的数据负载平衡而且保证相邻的GRID块存储在不同的处理结点上，使处理结点间的工作负载平衡，有效地支持各种数据操作. 最后，它使用顺序GRID文件结构组织单处理结点上的数据，其结果称为物理GRID文件. 为了保证数据均匀地分布在各个处理结点，他们给出了三个并行数据划分算法. 由于这三个算法具有很高的效率，它们既可用于初始的数据划分，也可以用于数据的再分布.

第一个算法是一个并行动态规划算法(PD). 对于数据划分问题，PD能够给出优化解. 这个算法的复杂性仅与F的各属性的值域大小有关，与文件的大小无关. 对于具有小值域属性的文件，这个算法具有很高的效率. 给定一个文件F，PD分两步完成F的划分. 第一步，PD使用多个处理结点，根据F中记录值的分布情况，基于优化原理，建立一组递归方程，并行地求解建立F的逻辑GRID文件的最优方案，实现F的划分，在多处理结点间分布F的逻辑GRID块. 第二步，对于$1\leqslant i\leqslant N$，处理结点i根据F_i(F_i是F在处理结点i上的子集合)中记录值的分布情况，基于优化原理，建立一组递归方程，并行地求解建立F_i的顺序GRID文件的最优方案，最后形成F的物理GRID文件.

第二个算法是一个动态规划和随机方法相结合的算法(PRD). PRD能够以很高的概率给出优化或近似优化解. 当文件的各属性的值域较大时，PRD效率高于PD. 给定文件F，PRD也分两步完成F的划分. 第一步，首先随机地从F中抽取样本集合S；然后，使用排序方法划分S的各维值域，形成划分F的方案，建立F的逻辑GRID文件；最后，完成F在多处理结点间的分布. 第二步类似于PD算法的第二步.

第三个算法是一个纯粹的随机算法(PFR). 对于数据划分问题，PFR能够以较高的概率给出优化或近似优化解. 当文件的各属性的值域很大时，PFR的复杂性和响应时间小于PRD. 给定文件F，PFR也分两步完成F的划分. 第一步类似于PRD算法的第一步. 第二步，对于$1\leqslant i\leqslant N$，处理结点i，随机地从F_i(F_i是F在处理结点i上的子集合)中抽取样本集合S_i，并使用排序的方法划分S_i的各个属性的值域，形成F_i的划分方案，建立F_i的顺序GRID文件，最后形成F的物理GRID文件.

参考文献

1 Hurson A R, Miller L L, Pakzad S H. Parallel architectures for database systems. New York: IEEE Computer Society Press, 1989.

2 Frieder O. Multiprocessor algorithms for relational database operators on hypercube systems. IEEE Computer, 1990(9):13—28.

3 Stonebraker M. The case for shared nothing. Database Eng., 1986,9(1):17—24.

4 李建中. 并行关系数据库系统的并行查询计划模型. 第10届中国数据库学术会议论文集，沈阳，1992，沈阳：东北工学院出版社，1992:58—64.

5 DeWitt D *et al*. The gamma database machine project. IEEE Trans. on Knowledge and Data Eng., 1990,2(1):44—62.

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

7 Tandem Database Group. NonStop SQL, a distributed, high—performance, high—reliability implementation of SQL, Workshop on High Performance Transaction Systems, Asilomar, CA, 1987.

8 Copeland G *et al*. Data placement in bubba. In: Larson P ed. Proc. of ACM SIGMOD'88,USA, 1988, Baltimore: ACM Press, 1988:99—108.

9 Ghandeharizadeh S, DeWitt D. Hybrid—range partitioning strategy: a new declustering strategy for multiprocessor database machines. In: McLeod D, Sacks—Davis R, Schek H eds. Proceedings of VLDB'90, Brisbane, 1990, Palo Alto: Morgan kaufmann Publishers, Inc., 1990:481—492.

10 Li Jianzhong, Srivastava J, Rotem D. CMD: A multi—dimensional declustering method for parallel database systems. In: Yuan Liyan ed. Proceedings of VLDB'92, VanCouver, 1992, San Mateo: Morgan kaufmann Publishers, Inc., 1992:1—14.

11 Niccum T M, Li Jianzhong. A parallel join algorithm for shared—memory database systems. In: Wu Jianping *et al*. eds. Proceedings of 1993 International Yuang Computer Scientist Conference,Beijing, 1993, Beijing:Tsinghua University Press, 1993.

12 Li Jianzhong. Range query procession in multidisk systems. Journal of Computer Science and Technology,1992,7(4):316—327.

13 Ghandeharizadeh S, DeWitt D. A performance analysis of alternative multi—attribute declustering strategies. In: Stonebraker M ed. Proceedings of ACM SIGMOD'92, San Diago CA, 1992, Baltimore: ACM Press, 1992:29—38.

14 Seeger B, Larson P A. Multi—disk b—tree. In: Clifford J, King Roger eds. Proceedings of ACM SIGMOD'91, Denver Colorado, 1991, Baltimore: ACM Press, 1991:436—445.

15 Hua K A, Lee Chiang. An adaptive data placement scheme for parallel database computer systems. In: McLeod D, Sacks—Davis R, Schek H eds. Proceedings of VLDB'90, Brisbane, 1990, Palo Alto: Morgan kaufmann Publishers, Inc., 1990:493—506.

16 Li Jianzhong, Rotem D, Srivastava J. Algorithms for loading parallel grid files. In: Buneman P, Jajodia S eds. Proceedings of ACM SIGMOD'93, Washington DC, 1993, Baltimore: ACM Press, 1993:347—356.

PARALLELIZATION TECHNIQUES FOR QUERY PROCESSING AND DATA DECLUSTERING METHODS

Li Jianzhong

(*Information Research Institute, Heilongjiang University, Harbin* 150080)

Abstract With the advent of parallel computer systems, parallel database systems have rapidly become an attractive research area in recent years. A number of parallel database projects have been started in academia and industry, and issues ranging from architecture to algorithms are being investigated. This paper is the first one of the two series papers that survey the research area of parallel database systems. In the paper, the current research directions of parallel database systems are surveyed, and physical design methods for parallel databases and parallelization techniques for query processing are reviewed as well.

Key words Parallel database, data declustering, data flow.