一种并行处理多维连接和聚集操作的有效方法

薛永生[1] 黄震华[1] 段江娇[1] 张延松[1] 吕晓华[2]
[1](厦门大学计算机科学系 厦门 361005)
[2](浙江理工大学信息电子学院 杭州 310018)
(jukie @netease. com)

摘 要 随着并行计算算法的完善和廉价、功能强大的多处理机系统的成熟,使得采用多处理机系统来并行处理多维数据仓库的连接和聚集操作成为当前有效提高 OLAP 查询处理性能的首选技术. 为此,提出一种降低连接和聚集操作开销的并行算法 PJAMDDC(parallel join and aggregation for multi-dimensional data cube). 算法充分考虑了多维数据立方体的存储机制和多处理机分布系统的结构特点,在原有聚集计算多维数据立方体的搜索点阵逻辑结构的基础上,采用多维数据仓库的层次联合代理(hierarchy combined surrogate)和对立方体的搜索点阵进行加权的方法,使得立方体数据在多个处理机间的分配达到最佳的状态,从而在分割多维数据的同时,提高了并行处理多维连接和聚集操作的效率. 算法实验评估表明,PJAMDDC 算法并行处理多维数据仓库的连接和聚集操作是有效的.

关键词 数据仓库;OLAP;层次联合代理;并行聚集查询;数据立方体

中图法分类号 TP311. 13

An Efficient Method for Parallel Multi-Dimensional Join and Aggregation

XUE Yong-Sheng[1], HUANG Zhen-Hua[1], DUAN Jiang-Jiao[1], ZHANG Yan-Song[1], and LÜ Xiao-Hua[2]
[1] *(Department of Computer Science, Xiamen University, Xiamen* 361005*)*
[2] *(College of Information and Electron, Zhejiang University of Sciences, Hangzhou* 310018*)*

Abstract Along with the perfection of parallel computing algorithms and the maturity of cheap but powerful multiprocessing systems, parallel multi-dimensional join and aggregation on data warehouse with the multiprocessing system becomes the preferred technology for efficiently improving OLAP operation. In this paper, a new method(PJAMDDC algorithm) is proposed for processing time-consuming join and aggregate operation. In this method, by taking into consideration the characteristics of the storage mechanism of multi-dimensional data cube and the structural characteristic of multiprocessing system and adopting hierarchy combining surrogate/ search lattice of data cube with weight on the basis of the former logic structure of search lattice of data cube. It advances the join and aggregate efficiency by achieving the optimal state for distribution of data cube between processors. As illustrated by experimental result of the performance analysis, PJAMDDC algorithm is efficient when it is used for parallel multi-dimensional join and aggregation on data warehouse.

Key words data warehouse; OLAP; hierarchy-combined surrogate; parallel aggregate query; data cube

1 引言与相关工作

通过对数据仓库中的低粒度数据的预聚集处理来生成高效的物化视图是联机分析处理(OLAP)的一个重要技术,而 OLAP 操作一般都是涉及大量数据的即席复杂查询[1]. 用户通过提交 OLAP 查询对数据进行分析,辅助决策,通常需要较快的查询响应

收稿日期:2004-07-15
基金项目:福建省自然科学基金项目(A0310008);福建省高新技术研究开放计划重点项目(2003H043)

速度. 近年来,随着并行计算算法的完善和廉价而功能强大的多处理机系统的成熟,使得采用多处理机系统来并行处理多维数据仓库的连接和聚集操作成为当前有效提高 OLAP 查询处理性能的首选技术. 目前主要有 MOLAP 和 ROLAP 两种方式可用于数据立方体的连接和聚集操作的实现[2]. 在 ROLAP 模式下,具有 K 个维的数据立方体被存储为 2^K 个关系表,每一张表存储着某一特定的立方体结构. 当进行数据立方体的连接和聚集操作时,就可以利用成熟和强大的关系数据库模式;并且对容量巨大的和数据稀疏的数据仓库更加有效. 所以本文就选择在 ROLAP 模式下并行处理多维数据立方体的连接和聚集操作. 并行处理多维连接和聚集操作的一个重要环节就是数据在各处理机间进行分割,而目前,在研究数据分割时所采用的方法大致有两种:基于排序的方法和基于 Hash 定址的方法[3]. 文献[3～4]给出了单个分割属性的并行计算方法,然而它未能对多维连接和聚集操作进行全局优化. 文献[5～6]用同一个算法框架结构描述了 5 种全局优化策略:①最小父结点法;②缓冲区排序法;③分步扫描法;④共享磁盘排序法和⑤共享磁盘分割法,并对多维数据立方体的并行查询进行了有效地全局优化. 文献[7]给出了多维立方体查询计算的迭代算法,该方法把全局聚集操作化分成若干子聚集操作,同时搜索执行这些子聚集操作所需的最小排序步骤. 并且通过对不同立方体的迭代计算来有效利用内存空间,从而极大地减少了磁盘访问的开销. 文献[8]充分考虑了海量稀疏立方体的结构特点,给出具有该结构的数据立方体的连接和聚集操作的递归算法. 该方法在处理复杂连接和聚集操作时基于两个基本思想:把数据表分割成适合内存大小的数据块和每个数据块独自执行连接和聚集操作. 然而这些算法按分组属性进行连接和聚集操作时考虑了所有的维表属性. 文献[9～10]改进了上面算法的不足,给出算法来预先判断和索引所用到的分组属性. 本文从优化多维数据立方体在多个处理机间的分组和连接性能出发,提出了一种新的并行处理多维数据立方体的连接和聚集操作算法——PJAMDDC(parallel join and aggregation for multi-dimensional data cube).

PJAMDDC 算法充分考虑了多维数据立方体的存储机制和多处理机分布系统的结构特点,在原有聚集计算多维数据立方体的搜索点阵逻辑结构[11]的基础上,采用多维数据仓库的层次联合代理(hierarchy combined surrogate)和对立方体的搜索点阵进行加权的方法,使得立方体数据在多个处理机间的分配达到最佳的状态,从而在分割多维数据的同时,提高了并行处理多维连接和聚集操作的效率.

文章接下来的部分是这样组织的:第 2 节给出多维数据仓库的层次联合代理方法;第 3 节给出多维数据立方体的搜索点阵的描述;第 4 节描述 PJAMDDC 算法;第 5 节进行 PJAMDDC 算法实验结果评估;最后对全文进行总结.

2 多维数据仓库的层次联合代理

在文献[12～14]中研究并提出了基于某一具体维的层次联合代理,然而 OLAP 操作通常要结合多个维的属性,所以我们必须将某一具体维的层次联合代理扩展为能够适用于多个维的情况.

2.1 有关多维数据仓库层次的若干定义

定义 1. 多维数据仓库层次的一棵层次树 H-Tree 是一个以 ALL 为根结点的 DAG(directed acyclic graph),可用二元组 $\Gamma = (\mathbb{V}, \partial)$ 表示. 其中 $\mathbb{V} = \{ALL, \mathbb{V}_1, \mathbb{V}_2, \cdots, \mathbb{V}_n\}$ 是 Γ 中结点集合,$\partial = \{\partial_{ij} | \partial_{ij}$ 表示 Γ 中有 $\mathbb{V}_i \rightarrow \mathbb{V}_j\}$ 是 Γ 中有向连线集合.

定义 2. 设维 K 的值域为 $R = \{\vartheta_1, \vartheta_2, \cdots, \vartheta_t\}$,对应层次树 H-Tree 的深度记为 γ,则它有 $\gamma + 1$ 层的有序集族,记为 $\prod = \{\xi^0, \xi^1, \cdots, \xi^\gamma\}$. 如果 $\lambda = (\chi_1, \chi_2, \cdots, \chi_m)$ 满足下列条件,则称 λ 为层次树 H-Tree 的第 i 层$(0 \leq i \leq \gamma)$ ξ^i 的成员组:

① $depth(\chi_j) = i;\ (1 \leq j \leq m)$

② $\chi_j \subseteq R$;

③ $\xi^i = \cup 0 \leq j \leq m \chi_j$;

④对 $\forall \chi_p, \chi_q \in \xi^i$ 且 $\chi_p \neq \chi_q$,则 $\chi_p \cap \chi_q = \varnothing$,

其中,$depth(\chi_j)$ 为 χ 的深度,第 i 层的第 j 个成员 $(1 \leq j \leq m)$ 简记为 χ_j^i. 显然,$R = \bigcup_{0 \leq i \leq \gamma, 1 \leq j \leq m} \chi_j^i$. 由定义 2 得知,处于同一层次上的各成员所表示的实体集不相互重叠.

定义 3. 成员 χ_j^i 的子成员集定义为

$$children(\chi_j^i) = \{\chi_l^{i+1} \in \xi^{i+1} | \chi_l^{i+1} \subseteq \chi_j^i\}.$$

定义 4. 成员 χ_j^i 的父成员集定义为

$$parent(\chi_j^i) = \{\chi_k^{i-1} \in \xi^{i-1} | \chi_k^{i-1} \supseteq \chi_j^i\}.$$

定义 5. 设 $|children(\chi_j^i)| = \tau$,则定义双射函

数 $BOrd_{\chi}$ 为

$children(\chi_j^i) \rightarrow \{0,1,\cdots,\tau-1\}$. 双射函数 $BOrd_{\chi}$ 为成员 χ_j^i 的每个子成员 $\chi_l^{i+1} \in children(\chi_j^i)$ 赋予一个互不相同的有序码位,从而定义了一种编码模式.

2.2　多维层次的联合代理

为了有效地对多维数据仓库的层次进行编码,减少并行处理多维连接和聚集操作的时间和空间消耗,我们采取了联合代理的方法.

定义 6. 深度为 γ 的层次树 H-Tree 上的 $\gamma+1$ 个有序集为 $\xi^0,\xi^1,\cdots,\xi^{\gamma},\xi^i(0\leq i\leq\gamma)$ 的 m 个成员记为 $\chi_1^i,\chi_2^i,\cdots,\chi_m^i$,赋予成员 χ_j^i 的子成员集 $children(\chi_j^i)$ 的双射函数为 $BOrd_{\chi_j^i}$,成员 χ_j^i 的代理值 $f(H,\chi_j^i)$ 与其父成员 χ_k^{i-1} 的代理值 $f(H,\chi_k^{i-1})$ 之间的连接记为 $\oplus$. 我们可用递归形式来定义在层次树 H-Tree 上各成员的联合代理值:

$$f(H,\chi_j^i)=\begin{cases} BOrd_{parent(\chi_j^i)}(\chi_j^i), & \text{if } i=1,\\ f(H,parent(\chi_j^i))\ \oplus BOrd_{parent(\chi_j^i)}(\chi_j^i), & \text{if } i\neq 1.\end{cases}$$

引理 1. 多维层次上的每个成员的联合代理值存在且惟一.

证明. 由定义 5 知,成员 χ_j^i 根据双射函数 $BOrd_{\chi_j^i}$ 得到的子成员集 $children(\chi_j^i)$ 中的各成员 $\chi_1^{i+1},\chi_2^{i+1},\cdots,\chi_{\tau}^{i+1}$ 的编码是有序且惟一的,进而由定义 6 知,从根成员到各个结点成员的编码联结也是惟一的,即得每个成员的联合代理值存在且惟一.

定义 7. 深度为 γ 的层次树 H-Tree 上的 $\gamma+1$ 个有序集为 $\xi^0,\xi^1,\cdots,\xi^{\gamma},\xi^i(0\leq i\leq\gamma)$ 的 m 个成员记为 $\chi_1^i,\chi_2^i,\cdots,\chi_m^i$,成员 χ_j^i 的子成员集为 $children(\chi_j^i)$,令 $\nabla(H,i+1)=\mathrm{Max}\{Card(children(\chi_j^i))\mid \forall\chi_j^i\in\xi^i\}$,其中 $Card$ 为取集合容量的函数. 则我们定义第 $i+1$ 层的层跨距为

$$\in^{i+1}=\left\lceil \log_2 \nabla(H,i+1)\right\rceil.$$

据定义 7 知,若成员 χ_p,χ_q 隶属层次树 H-Tree 的同一层 ξ^i,那么它们所对应的层跨距 $\in^i$ 必相等,从而成员 χ_p 和 χ_q 的编码长度是一致的. 因此,可以用一种压缩的二进制编码方式进行统一管理各成员的联合代理值.

定义 8. 设路径 Φ 遍历层次树 H-Tree 的 t 个层的成员 $\chi^1,\chi^2,\cdots,\chi^t$,赋予成员 $\chi^i(1\leq i\leq t)$ 的子成员集 $children(\chi^i)$ 的双射函数为 $BOrd_{\chi^i}$,则定义路径 Φ 的联合代理值为

$$f(H,\Phi)=f(H,\chi^t)=BOrd_{parent(\chi^1)}(\chi^1)+BOrd_{parent(\chi^2)}(\chi^2)\cdot 2^{\in^i}+\cdots+BOrd_{parent(\chi^t)}(\chi^t)\cdot 2^{\in^i+\in^2+\cdots+\in^{t-1}}.$$

当以定义 6～8 来编码存储多维数据仓库层次树 H-Tree 上的各结点时,优点在于它能够用较少并且统一的位数来存储较多的数据,并且减少搜索满足条件的记录的时间开销,从而提高多维连接和聚集操作的效率.

3　多维数据立方体的搜索点阵

定义 9. 多维数据立方体的搜索点阵[11]是一个有向非循环图 (directed acyclic graph) $J=(\Gamma,E)$. 每个结点 $\zeta\in\Gamma$ 表示某一粒度和聚集属性的立方体;每条边 $\langle\omega,\zeta\rangle\in E$ 表示立方体 ω 能够计算产生出立方体 ζ. 结点 ω 称做结点 ζ 的父结点,且结点 ζ 比结点 ω 少一个聚集属性,因此它的粒度更为粗糙.

由定义 9 知,具有 K 个维的数据仓库表示成多维数据立方体的搜索点阵时,它有 K 个结点.

我们约定在 K 个维的数据立方体的搜索点阵中,如果某个结点具有 $\zeta(\zeta\leq K)$ 个聚集属性,则它处于第 $K-\zeta+1$ 层. 图 1 显示了基于具有 4 个维属性 (A,B,C,D) 的关系数据仓库 $\hat{W}$ 的搜索点阵,每个结点都标上相应的维子集.

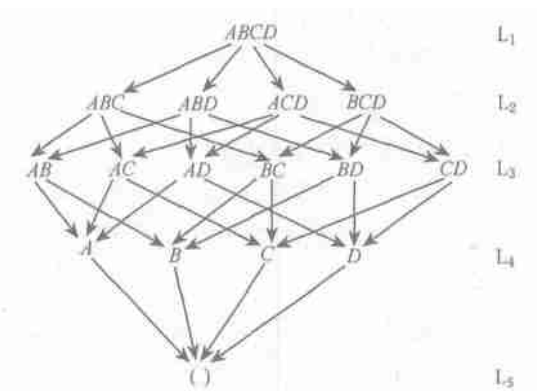

图 1　基于具有 4 个维属性的关系数据仓库的搜索点阵

从图 1 可以看出,关系数据仓库 $\hat{W}$ 具有 $2^4=16$ 个不同的维聚集立方体,并且每个维聚集立方体都对应关系数据仓库 $\hat{W}$ 中的一个关系表.

定义 10. 设 $J = (\Gamma, E)$ 是加权的有向连通图，$\sigma_1, \sigma_2, \cdots, \sigma_p$ 是图 J 的 p 棵有向生成树. $\sigma_j (1 \leq j \leq p)$ 中树枝的权之和称为 σ_j 的权，记为 $\Psi(\sigma_j) = \sum_{e_{jk} \in \sigma_j} h(e_{jk})$，$h(e_{jk})$ 为树枝 e_{jk} 的权. 如果 $\Psi(\sigma_t) = \mathrm{MIN}(\{\Psi(\sigma_1), \Psi(\sigma_2), \cdots, \Psi(\sigma_p)\})$，则我们称 $\sigma_t (1 \leq t \leq p)$ 为图 J 的有向最小生成树.

目前可用于求最小生成树的算法有 3 种，分别为 KRUSKAL 算法[15]、管梅谷算法和 GREEDY 算法.

4 PJAMDDC 算法描述

近年来，人们研究连接和聚集算法的方法可分为两类：

(1) 聚集查询的并行处理和结合聚集操作的查询优化方法，其中有代表性是文献[16,17].

(2) 从优化 OLAP 操作的本身出发来研究聚集算法，其中有代表性是文献[18]. 由方法(1)，(2)得到的连接和聚集算法只能够优化某一方面，有些算法可以有效提高连接的效率，但是维表本身的搜索开销很大；有些算法可以快速得到满足条件的结果集，但是它只基于单维的，从而限制了应用的范围.

本文从优化多维数据立方体在多个处理机间的分组和连接性能出发提出了一种新的并行处理多维数据立方体的连接和聚集操作算法——PJAMDDC. PJAMDDC 算法类似于文献[19]中所采用的最小父结点的并行处理方法，包括两个阶段的工作. 在第 1 阶段，我们构建用于评估数据立方体连接和聚集计算的模型，它是一棵由加权的多维数据立方体的搜索点阵演变而来的加权有向最小生成树. 在第 2 阶段我们把多个处理机分配给由第 1 阶段得到的有向最小生成树的结点组，让处理机系统并行对各结点集合执行连接和聚集操作.

4.1 构建加权有向最小生成树

由第 3 节我们知道，可以通过构建多维数据立方体的搜索点阵来连接和聚集计算基于 ROLAP 模式下的多维数据立方体. 在本小节，我们给搜索点阵的每条边赋上某一权值，该权值的大小等于父结点产生子结点的连接和聚集计算的开销. 接下来，我们在该加权搜索点阵的基础上运用 KRUSKAL 算法来生成加权有向最小生成树. 但是我们事先不能精确计算出加权最小生成树上的各权值，所采取的方法是：多维数据立方体的关系模式用多维数据仓库的层次联合代理的编码文件来存储，并且从磁盘数据页中随机选取一小部分数据作为采样数据；通过使用一些有效的采样评估算法[20]来近似计算每一条边的权值. 所以在 PJAMDDC 算法的第 1 阶段我们构建一个评估加权有向最小生成树 σ，σ 上的各边权值均是近似精确的评估值.

下面我们给出 PJAMDDC 算法第 1 阶段的处理步骤：

输入：多维数据仓库的层次联合代理编码文件 ρ，r 个处理机的集合 $\{\Omega_1, \Omega_2, \cdots, \Omega_r\}$，分组属性 $GA_1, \cdots, GA_s$；

输出：各边权值近似精确的评估加权有向最小生成树 σ.

(1) 根据初始查询 Q 的分组属性 $GA_1, \cdots, GA_s$，将多维数据仓库的层次联合代理编码文件 ρ 水平分割成 r 个子编码文件 $\rho^1, \rho^2, \cdots, \rho^r$；

(2) For $j = 1$ to r

① 将编码文件 ρ_j 分配给处理机 Ω_j；

② Ω_j 为编码文件 ρ^j 构建多维数据立方体的搜索点阵 $J_j = (\Gamma_j, E_j)$，并给每一条边赋上某一权值，权值的获得方法是：从存储编码文件 ρ^j 的磁盘数据页中随机选取一小部分数据作为采样数据；通过使用一些有效的采样评估算法[20]来近似计算每一条边的权值；

(3) 从第(2)步得到的 r 个多维数据立方体的搜索点阵 $J_1, J_2, \cdots, J_r$，构建相对全局编码文件 ρ 的多维数据立方体搜索点阵 $J = (\Gamma, E)$，其中每条边的加权赋值如下：$h(e_k) = \left\lceil \sum_{i=1}^{r} e_{ik} / r \right\rceil$；

(4) 由全局加权搜索点阵 $J = (\Gamma, E)$，运用 KRUSKAL 算法[15]来生成加权有向最小生成树 σ.

4.2 处理机分配策略

PJAMDDC 算法的第 2 阶段是在第 1 阶段产生的加权有向最小生成树 σ 的基础上，把处理机分配给树上的各个结点，使得计算各结点的时间开销最小，从而使得多维立方体的连接和聚集计算的时间开销降到最低. 然而，在处理机的分配过程中，有两个因素对总的时间开销影响很大：①处理机怎样分配给加权有向最小生成树 σ 的各个结点；②要分配多少处理机给树的各结点.

假设多处理机系统有 r 个可用处理机，首先我们把这 r 个处理机分配给加权有向最小生成树 σ 的根结点 R. 假定 R 有 μ 个子结点，分布记为 $\omega_1, \omega_2, \cdots, \omega_\mu$，以 $\omega_j (1 \leq j \leq \mu)$ 为根结点的子树记为 V_j，相应的权值为 $\Psi(V_j) = \sum_{e_{jk} \in V_j} h(e_{jk})$. 并且结点 $\omega_j (1 \leq$

$j \le \mu$)的连接和聚集计算要在其父结点 R 计算完成之后才能进行. 连接和聚集计算这些以 $\omega_1, \omega_2, \cdots, \omega_\mu$ 为根结点的 $\mathcal{F}$ 棵子树所花费的最小时间开销为 $T_{\min} = \left\lceil \sum_{j=1}^{\mu} \Psi(V_j) / r \right\rceil$. 为了方便而不失一般性,我们假设 $\Psi(V_1) \ge \Psi(V_2) \ge \cdots \ge \Psi(V_\mu)$,并且扫描子树的顺序就按这个次序.

规则 1. 对于以结点 ω_j 为根结点的子树 V_j ($1 \le j \le \mu$),如果 $\left\lfloor \Psi(V_j) / T_{\min} \right\rfloor \ne 0$,那么分配 $\left\lfloor \Psi(V_j) / T_{\min} \right\rfloor$ 个处理机给子树 V_j;否则合并子树 V_j 到子树 V_{j+1} 中,并重复执行该操作,直到该值不为 0,从而有若干个处理机分配给它.

引理 2. 根据规则 1 我们分配给子树 $V_1, V_2, \cdots, V_{\mathcal{F}}$ 的处理机总个数记为 γ,实际多处理机系统的可用处理机总个数记为 r,则 $\gamma \le r$.

证明.

$$\gamma = \sum_{j=1}^{\mu} \left\lfloor \Psi(V_j) / T_{\min} \right\rfloor \le \sum_{j=1}^{\mu} \Psi(V_j) / T_{\min} = \sum_{j=1}^{\mu} \Psi(V_j) / \left\lceil \sum_{j=1}^{\mu} \Psi(V_j) / r \right\rceil \le \sum_{j=1}^{\mu} \Psi(V_j) / \sum_{j=1}^{\mu} \Psi(V_j) / r = r.$$

由引理 2 可知,按照规则 1,我们的处理机分配策略不会出现所需的处理机总数超过实际的多处理机系统的可用处理机总数的情况. 从而说明了本小节所提出的处理机分配策略在处理机的数量满足度上是可行的.

下面我们给出 PJAMDDC 算法第 2 阶段的处理步骤:

```
PJAMDDC-Second( σ_R, {Ω_1, Ω_2, ···, Ω_r})
  /* σ为第 1 阶段产生的加权有向最小生成树; R为树 σ的根结点; {Ω_1, Ω_2, ···, Ω_r}为
     r 个多处理机系统实际可用的处理机 */
{
  Ξ(R) = Processing-root(R, {Ω_1, Ω_2, ···, Ω_r});
    /* 分配处理机系统{Ω_1, Ω_2, ···, Ω_r}给根结点 R,计算出 R 的时间开销 */
  T_min = ⌈ (Ψ(σ - Ξ(R)) / r ⌉;
  △(R) = Children(R) = {ω_1, ω_2, ···ω_μ};
    /* {ω_1, ω_2, ···ω_μ}为根结点 R 的子结点集合 */
  For j = 1 to μ
  {
    ∇(ω_j) = Producing-tree(ω_j) = V_j;
  };
  q = 0;
  repeat
    q = q + 1;
    g = ⌊ Ψ(V_q) / T_min ⌋;
    If (g ≠ 0)
    {
      Assigning-processors(g, V_q);
        /* 分配 g 个处理机给子树 V_q */
      If (g > 1)
      {
        PJAMDDC-Second(V_q, ω_q, {Ω_1, Ω_2, ···, Ω_g});
          /* {Ω_1, Ω_2, ···, Ω_g}为 q 个处理机的集合 */
      }
    }
    Else
    {
      q' = q;
      Ψ' = Ψ(V_q);
      While((g = 0) and (q ≠ μ))
      {
        q = q + 1;
        Ψ' = Ψ' + Ψ(V_q);
        g = ⌊ Ψ' / T_min; ⌋;
      };
      Assigning-processors(1, {V_q', V_q'+1, ···, V_q});
      Run APM algorithm[21] to compute {V_q', V_q'+1, ···, V_q};
    }
  Until q = μ;
}
```

引理 3. 在 PJAMDDC 算法的第 2 阶段处理过程中,如果 while 循环被执行,那么当该循环结束时,$g = 1$ 或者 $g = 0$,并且 $g = 0$ 当且仅当 $q = \mu$.

证明.

(1) $g = 0$ 当且仅当 $q = \mu$ 的情况;

①若 $g = 0$,根据前提可知,while 循环被执行,所以进步循环时 $q \ne \mu$;当要退出循环,则 $(g = 0)$ 和 $(q \ne \mu)$ 两个条件必须有一个不满足,所以 $q = \mu$.

②同理可知,若 $q = \mu$,则 $g = 0$.

(2) $g = 1$ 的情况;

假定当算法退出 while 循环时,包括 δ 棵子树,则 $\delta = q - q' + 1$. 当 $\delta = 1$ 时,

$$g = \left\lfloor \Psi / T_{\min} \right\rfloor = \left\lfloor (\Psi(V_q) + \Psi(V_{q'+1})) / T_{\min} \right\rfloor,$$

又根据前面的假设 $\Psi(V_{q'+1}) \leq \Psi(V_q)$,所以知:

$$\left\lfloor (\Psi(V_q) + \Psi(V_{q'+1})) / T_{\min} \right\rfloor \leq \left\lfloor 2\Psi(V_q) / T_{\min} \right\rfloor,$$

又因为,$0 < \Psi(V_q) / T_{\min} < 1$,所以 $0 < \Psi / T_{\min} < 2$,从而知 $g = \left\lfloor \Psi / T_{\min} \right\rfloor = 1$. 当 $\delta > 1$ 时,根据循环条件知,$0 < \sum_{j=q}^{q-1} \Psi(V_j) / T_{\min} < 1$,又根据前面的假设 $\Psi(V_q) \leq \Psi(V_{q-1})$. 所以,$0 < \Psi / T_{\min} = \sum_{j=q}^{q} \Psi(V_j) / T_{\min} < 2 \sum_{j=q}^{q-1} \Psi(V_j) / T_{\min} < 2$,从而得:$0 \leq g = \left\lfloor \Psi / T_{\min} \right\rfloor < 2$,因为排除 $g = 0$ 的情况(该情况见证明(1)),所以 $g = 1$.

5 算法实验评估

我们在并行多维数据立方体的连接和聚集操作的研究中,实现了 PJAMDDC 算法,并进行了算法实验的评估. 实验用的硬件设备是无共享内存的集群系统,该系统里有 9 个处理机,每个处理机配有 512MB 的内存和 80GB 的磁盘;图 2 显示了该集群系统的结构.

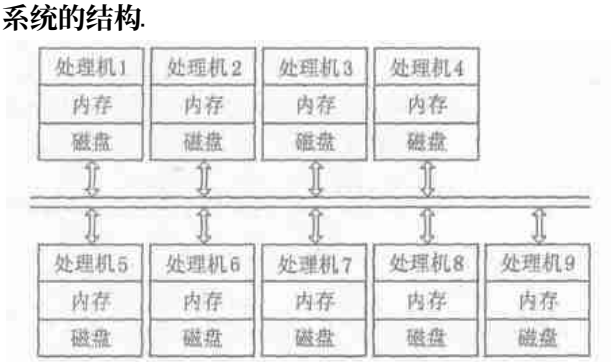

图 2 无共享内存的集群系统

数据库使用的是 Oracle9i 系统. 实验中用到的数据是我们通过人为构建的数据仓库,其中包括 4 个维表(*Time*, *Part*, *Supplier*, *Customer*)和 1 个事实表(*Sales*). 图 3 给出了我们所构建的数据仓库中维表和事实表的结构. 用于对该数据仓库进行 OLAP 聚集查询的 SQL 语句如图 4 所示. 并且实验中用到的 4 个维表和 1 个事实表的部分数据如表 1 至表 5 所示.

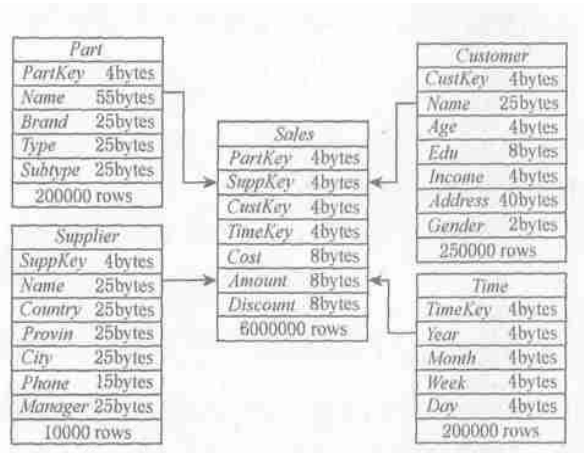

图 3 人为数据仓库的表结构

```
SELECT   P. brand, Su. name, C. income, T. month, Sun(S. cost),
         AVG(C. incom)
FROM     Part p, Supplier Su, Customer C, Time T, Sales S
WHERE    (P. Part Key = S. Part Key)
         AND(Su. Supp Key = S. Supp Key)
         AND(C. Cust Key = S. Cust Key)
         AND(T. Timekey = S. Time Key)
         AND(P. type = 'Food')
         AND(Su. manager = 'Smith')
         AND(C. edu = 'College')
GROUP BY P. brand, Su. name, C. income, T. month
HAVING   AVG(C. income) > 2000
```

图 4 具有聚集操作的 OLAP 查询

表 1 时间维部分数据

Time Key	Day	Month	Week	Year
344	21	3	12	1999
443	3	8	32	2002
123	12	4	15	2001
913	27	11	44	2003
…	…	…	…	…

表 2 产品维部分数据

Part Key	Name	Brand	Subtype	Type
3	Kristiet	Markexs	Skirt	Clothing
7	Relox	Zduet	Jean	Clothing
21	Sweet Tooth	Chewy Industries	Candy	Food
11	Paomian	Kangshifu	Fastfood	Food
7	Paxit	Jeanes	Qixinshi	Cosmetic
…	…	…	…	…

表 3　代理商维部分数据

SuppKey	*Name*	*Country*	*Provin*	*City*	*Manager*	*Phone*
3	Tafrei	China	Fujian	Fuzhou	Lifei	xxxx-xxxxxxx
6	Jexecf	China	Jiangshu	Shuzhou	Huangyi	xxxx-xxxxxxx
7	Fucklex	China	Jiangxi	Nanchan	Smith	xxxx-xxxxxxx
2	Renex	China	Shandong	Yantai	Chenyan	xxxx-xxxxxxx
5	Uyerwa	China	Guizhou	Gui Yang	Yangkun	xxxx-xxxxxxx
…	…	…	…	…	…	…

表 4　顾客维部分数据

CustKey	*Name*	*Age*	*Gender*	*Edu*	*Income*	*Address*
5	Liyang	32	M	College	3700	xxxxxxxxxx
12	Chenli	25	M	High School	1370	xxxxxxxxxx
4	Xuxian	41	FM	College	5000	xxxxxxxxxx
24	Liuxiao	29	M	College	4100	xxxxxxxxxx
8	Wangjin	38	FM	High School	2000	xxxxxxxxxx
…	…	…	…	…	…	…

表 5　事实表部分数据

TimeKey	*PartKey*	*SuppKey*	*CustKey*	*Amount*	*Cost*	*Discount*
443	7	2	12	20	400	0.85
123	3	11	7	10	2100	0.9
443	21	7	8	150	6578	0.92
443	23	19	24	21	4100	0.74
8	1	12	10	2	98	0.88
18	5	3	33	7	200	0.9
3	12	4	2	90	9543	0.85
9	1	2	23	5	120	0.79
72	23	8	7	11	310	0.81
…	…	…	…	…	…	…

在实验中,我们分两种情况来评估算法性能.

(1) 由于评估多维数据仓库并行连接和聚集操作效率的一个重要指标是时间开销,所以在实验中,我们比较了 PJAMDDC 算法和当前流行的两个算法 ①PBJI 算法和 ②PHJ 算法[22]在处理图 4 所示的 OLAP 查询操作中的时间开销. 图 5 显示了当多处理机系统中可用处理机的个数从 1 个增加到 9 个时,3 个算法在运行时间(秒)上的变化情况.

(2) 当处理机个数不变时(我们实验用到的处理机个数为 9 个),我们评估了 PJAMDDC 算法和 PBJI 算法及 PHJ 算法分别在不同的数据仓库表记录数下的时间开销. 图 6 显示了 3 个算法的时间开销情况.

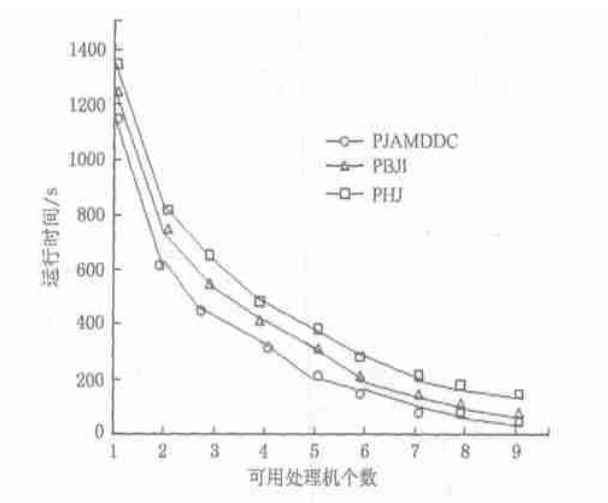

图 5　OLAP 查询运行时间随处理机个数的变化情况

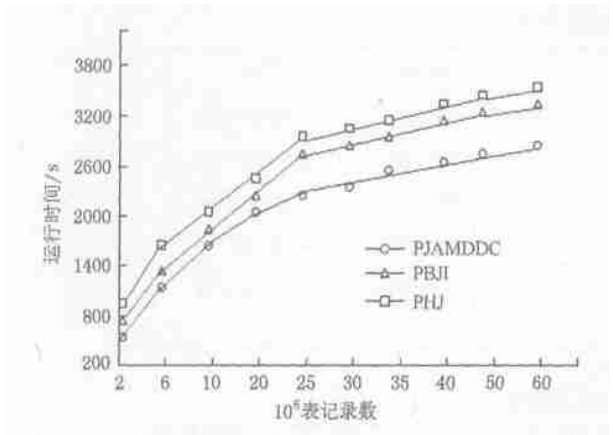

图 6 OLAP 查询运行时间随数据仓库记录数的变化情况

6 结 论

由于海量数据的存在,所以在多维数据仓库中执行复杂的连接和聚集操作的开销将会是很大. 随着并行计算算法的完善和廉价而功能强大的多处理机系统的成熟,使得采用多处理机系统来并行处理多维数据仓库的连接和聚集操作成为当前有效提高 OLAP 查询处理性能的首选技术.

本文充分考虑了多维数据立方体的存储机制和多处理机分布系统的结构特点,在原有聚集计算多维数据立方体的搜索点阵逻辑结构的基础上,采用多维数据仓库的层次联合代理和对立方体的搜索点阵进行加权的方法,使得立方体数据在多个处理机间的分配达到最佳的状态,从而在分割多维数据的同时,提高了并行处理多维连接和聚集操作的效率.

为了显示 PJAMDDC 算法的有效性和优越性,以 OLAP 查询操作时间开销的方式比较了 PJAMDDC 算法和目前流行的两个算法之间的代价差异并进行实验验证评估,从而得出了结论:PJAMDDC 算法比目前流行算法的效率高.

参 考 文 献

1 S Chaudhuri, U Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 1997, 26(1): 65～74

2 O Neil, P D Quass. Improved query performance with variant indexes. ACM SIGMOD Record, 1997, 26(2): 38～49

3 G Graefe. Query evaluation techniques for large databases. ACM SIGMOD Record, 1993, 25(1): 73～170

4 A Shatdal, J F Naughton. Adaptive parallel aggregation algorithms. In: Proc of the 1995 ACM SIGMOD Conf on Management of Data. New York: ACM Press, 1995. 104～114

5 L Cabibbo, R Torlone. Querying multidimensional databases. In: SCluet, RHull, eds. The 23th Int'l Conf on Computer Science, LNCS 1369. New York: Springer, 1997. 319～335

6 S Sarawagi, R Agrawal, A Gupta. On computing the data cube. IBM Almaden Research Center, Tech Rep: RJ10026, 1996

7 P M Deshpande, S Agarwal, J F Naughton, *et al*. Computation of multidimensional aggregates. Department of Computer Science, University of Wisconsin-Madison, Tech Rep: 1314, 1996

8 K A Ross, D Srivastava. Fast computation of sparse data cubes. In: Proc of the 23rd Int'l Conf on Very Large Data Bases. San Francisco: Morgan Kaufmann, 1997. 201～219

9 V Harinarayan, A Rajaraman, J D Ullman. Implementing data cubes efficiently. In: KBarker, U Manitoba eds. Proc of the 5th Int'l Conf on Information and Knowledge Management CIKM'96. New York: ACM Press, 1996. 205～216

10 H Gupta, V Harinarayan, A Rajaraman, *et al*. Index selection for OLAP. In: Proc of the 15th Int'l Conf on Int'l Council for Open and Distance Education. Los Alamitos, CA: IEEE Computer Society Press, 1997. 136～154

11 I S Mumick, D Quass, B S Mumick. Maintenance of data cubes and summary tables in a warehouse. ACM SIGMOD Record, 1997, 26(2): 132～141

12 C Li, X S Wang. A data model for supporting on-line analytical processing. In: KBarker, U Manitoba, eds. Proc of the 5th Int'l Conf on Information and Knowledge Management CIKM'96. New York: ACM Press, 1996. 81～88

13 E Bertino, W Kim. Indexing technique for queries on nested objects. IEEE Trans on Knowledge and Data Engineering, 1989, 1(2): 196～214

14 C Zou, B Salzberg, R Ladin. Back to the future: Dynamic hierarchical clustering. In: Proc of the 16th Int'l Conf on Int'l Council for Open and Distance Education. Oslo: SEAMEO, 1998. 578～587

15 S Xiaojun, H Qing. Efficient embedding k-ary complete trees into hypercubes parallel processing symposium. In: Proc of the 12th Int'l Conf on Data Engneering. Los Alamitos: IEEE Computer Society Press, 1996. 24～31

16 W Peng, L Per-Ake. Eager aggregation and lazy aggregation. In: D Umeshwar, *et al* eds. Proc of the 21st Int'l Conf on Very Large Data Bases. San Francisco: Morgan Kaufmann, 1995. 345～357

17 S Agarwal, R Agrawal, P M Deshpande. On the computation of multidimensional aggregates. In: T M Vijayaraman, A P Buchmann, C Mohan eds. Proc of the 22nd Int'l Conf on Very Large Data Bases VLDB'96. San Francisco: Morgan Kaufmann, 1996. 506～521

18 蒋旭东,冯建华. 联机分析查询处理中的一种聚集算法. 软件学报, 2002, 13(1): 65～70
(Jiang Xudong, Feng Jianhua. A novel aggregation algorithm for online analytical processing query evaluation. Journal of Software

(in Chinese), 2002, 13(1): 65～70)

19 J Gray, A Bosworth, A Layman, *et al*. Data cube: A relational aggregation operator generalizing group-by, cross-tab & sub-total. In: D Umeshwar, *et al* eds. Proc of the 21st Int'l Conf on Very Large Data Bases. San Francisco: Morgan Kaufmann, 1995. 345～357

20 F Olken, D Rotem. Simple random sampling from relational databases. In: Proc of the 12th VLDB. San Francisco: Morgan Kaufmann, 1986. 160～169

21 D Taniar, Y Jiang, K H Liu, *et al*. Aggregate-join query processing in parallel database systems. IEEE Trans on Knowledge and Data Engineering, 2000, 13(2): 824～829

22 A Datta, D VanderMeer, K Ramamritham. Parallel star join + dataIndexes: Efficient query processing in data warehouses and OLAP. IEEE Trans on Knowledge and Data Engineering, 2002, 14(6): 1299～1316

薛永生　男,1946 年生,教授,主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘等.

黄震华　男,1980 年生,硕士研究生,主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘等.

段江娇　女,1972 年生,硕士,讲师,主要研究方向为数据库理论与应用、分布式数据库、数据仓库、数据挖掘、网络技术等.

张延松　男,1973 年生,硕士研究生,讲师,主要研究方向为数据库、数据仓库和数据网格等.

吕晓华　男,1970 年生,硕士,讲师,主要研究方向为分布式数据库、数据仓库与数据挖掘、网络技术.