基于 Shared-Nothing 的并行 Hash 连接算法效率分析*

李庆华 睢海燕 邓冲

(国家高性能计算中心 武汉 430074)
(华中理工大学计算机学院 武汉 430074)
E-mail: nhpcc@public.wuhan.cngb.com

摘要 该文研究了基于 Shared-Nothing 结构的几种常用并行连接算法,分析了影响查询响应时间的各种因素.在此基础上,以多种硬件成分作为参数建立一个代价分析模型,使用该模型计算并行 Hash 算法在每个处理机上的平均任务执行时间和总的查询响应时间,并比较了几种算法在不同硬件配置下的执行效率.所提出的模型和分析方法为评价和选取并行连接算法提供了一种可行的途径.

关键词 查询并行处理,并行连接,查询响应时间,性能评价模型.
中图法分类号 TP338

多年来,并行连接算法一直是并行查询优化中的一个研究热点[1].人们在串行算法的基础上已提出了3种有效的并行连接算法:并行嵌套循环(nest-loop)连接算法、并行排序归并(sort-merge)连接算法和并行哈希(Hash)连接算法.大量文献表明[2],当处理相等连接、关系很大且内存足够大时,Hash 连接算法有明显的优势.由于其高效性和容易被并行化的特性,并行 Hash 连接算法的研究引起了人们的关注,相继提出了多种基于 Hash 的并行连接算法[3,4].

基于 SN(shared_nothing)结构,如何评价一个连接算法的性能,如何根据实际的应用选取连接算法,需要一个好的性能评价模型.然而,经多年研究,至今尚未提出一个能被广泛接受的、好的性能评价模型.

本文首先研究了基于 SN 结构的 Simple-Hash,Grace-Hash,Hybrid-Hash 这3种并行连接算法,分析了影响查询响应时间的各种因素,然后以多种硬件成分作为参数建立一个代价分析模型,使用该模型来计算并行 Hash 算法在每个处理机上的平均任务执行时间和查询响应时间,并分析了内存容量、网络传输速度、处理机数等硬件参数对这3种算法执行效率的影响,从而为如何评价和选取并行连接算法提供了一种可行的途径.

1 并行 Hash 连接算法

假定系统中有 P 个处理机,R 和 S 是作连接运算的两个关系,R 为其中较小的关系.这3种算法都采用散列水平划分法.R_i,S_i 分别表示处理机 i 上的 R 分片和 S 分片.实际的连接运算由对分块建造 Hash 表(building)和探测匹配元组(probing)实现.

1.1 并行 Simple-Hash-Join 算法

Step 1. 处理机 i($i=1,2,\ldots,p$)从本地磁盘读取 R_i,对其中每个元组的连接属性用 Hash 函数 h_1 进行散列,以决定其处理机号 j.如果 $j\neq i$,则将其放入本地的写盘缓冲区,否则放入相应的远程处理机的通信缓冲区中,当缓冲区满时,写回磁盘或向远程处理机发送.同时,处理机接收并合并从其余 $P-1$ 个处理机发送来的元组.在这一步结束时,关系 R 在各个处理机间重新分配.

---

* 本文研究得到国家863高科技项目基金(No.863-306-ZD-01-7)资助.作者李庆华,1940年生,教授,博士生导师,主要研究领域为并行处理与高性能计算,分布式系统,智能软件.睢海燕,女,1975年生,硕士生,主要研究领域为并行处理与高性能计算,分布式系统.邓冲,女,1972年生,工程师,主要研究领域为并行处理.

本文通讯联系人:李庆华,武汉 430074,华中理工大学计算机学院

本文 1998-10-08 收到原稿,1999-04-02 收到修改稿

Step 2. 以同样的 Hash 函数在各个处理机间重分配 S.

Step 3. 各个处理机分别定义自己的 Hash 函数 h_2,并行地做 Simple-Hash-Join. 最后进行结果合并.

1.2 并行 Grace-Hash-Join 算法

(1) 分配阶段

Step 1. 处理机 $i(i=1,2,\ldots,p)$从本地磁盘读取 R_i,对其中每个元组的连接属性用 Hash 函数 h_1 进行散列,以决定其处理机号 j 和桶号 $k(k=1,2,\ldots,B,B$ 与处理机数无关,只需保证每个桶的 Hash 表能够装入本地的内存). 若它属于本处理机的某个桶,则存入相应的写盘缓冲区,否则放入相应的远程处理机的通信缓冲区中. 当缓冲区满时,写回磁盘或向远程处理机发送. 同时,处理机接收并合并从其余 $P-1$ 个处理机发送来的桶. 在这一步结束时,关系 R 在各个处理机间重新分配.

Step 2. 以同样的 Hash 函数在各个处理机间重分配 S.

(2) 连接阶段

处理机 $i(i=1,2,\ldots,p)$ par-do

For $j=1$ to B 重复 Step 3,Step 4.

Step 3. 从磁盘读出 R'_{ij},创建 Hash 表.

Step 4. 从磁盘读出 S'_{ij},用每一个元组探测 R'_{ij} 的 Hash 表,输出结果.

1.3 并行 Hybrid-Hash-Join 算法

并行 Hybrid-Hash-Join 算法和并行 Grace-Hash-Join 算法的主要区别在于:Grace-Hash-Join 算法在第 1 步中利用多余的内存来存放第 1 个桶的文件,而 Hybrid-Hash-Join 算法则利用这片内存来构造第 1 个桶的 Hash 表. 因而在第 2 步 S 中属于第 1 个桶的元组可以立刻探测内存 Hash 表,而无需再保存于临时文件中. 两个算法都是由小关系的大小来决定桶的数量.

(1) 分配阶段

Step 1. 处理机 $i(i=1,2,\ldots,p)$从本地磁盘读取 R_i,对其中每个元组的连接属性用 Hash 函数 h_1 进行散列,以决定其处理机号 j,用函数 h_2 决定其桶号 $k(k=0,1,\ldots,B,B$ 与处理机数无关,只需保证每个桶的 Hash 表能够装入本地的内存). 若它属于本处理机且 $k=1$,则元组插入到内存的 Hash 表中;若属于本处理机但 $k\neq1$,则存入相应的写盘缓冲区;若不属于本处理机,则放入相应的远程处理机的通信缓冲区中. 当缓冲区满时,写回磁盘或向远程处理机发送. 同时,处理机接收从其余 $P-1$ 个处理机发送来的桶,用函数 h_2 决定其插入 Hash 表中或写入磁盘. 在这一步结束时,关系 R 在各个处理机间重新分配.

Step 2. 以同样的 Hash 函数划分 S_i,若属于本处理机的第 1 个桶,则将它与 R_{i1} 内存 Hash 表匹配连接;若属于本处理机的其余桶,则存入磁盘;若属于其他处理机,则发送到相应的处理机.

(2) 连接阶段

处理机 $i(i=1,2,\ldots,p)$ par-do

For $j=2$ to B 重复 Step 3,Step 4.

Step 3. 从磁盘读出 R'_{ij},创建 Hash 表.

Step 4. 从磁盘读出 S'_{ij},用每一个元组探测 R'_{ij} 的 Hash 表,输出结果.

2 算法的响应时间分析

在理想情况下,各个处理机承担相同的任务,具有相同的执行时间,此时,查询响应时间即等于任意一个处理机上的任务执行时间. 但是,由于受很多因素的影响,某些任务在完成到可以提交时存在一段时间间隔,称为同步延时. 导致算法同步延时的因素主要有数据偏斜(data skew),它包括元组放置偏斜、选择偏斜、重分布偏斜、乘积偏斜. 另外,还有一些随机因素,比如因数据放在不同磁盘的不同地方而导致不同的数据读取时间、通信网络的随机通信延迟等[5,6].

用 $Q(0\leqslant Q\leqslant 1)$ 表示数据分布偏斜.Q 表示关系中含最高偏斜属性值的一个片段,被分配到一个特定的桶.假定关系的剩余部分在属性值上的分布相对均匀.当处理机个数为 P 且 $Q\leqslant 1/P$ 时,可视为无偏斜,否则有偏斜.i 表示任一处理机号;k 表示接受偏斜桶的处理机号;j 表示不同于 k 的任一处理机号;T 表示查询响应时间;$T(i)$ 表示处理机 i 上的平均任务执行时间;$T(i)_{cpu}$ 表示处理机 i 上的 cpu 时间,由 cpu 处理元组时间,处理物理 I/O 时间和 cpu 处理机间通信时间组成;$T(i)_{disk}$ 表示处理机 i 上的磁盘传输时间;$T(i)_{net}$ 表示处理机 i 上的网络传输时间.

根据在第 1 节中所描述的算法,在流水线并行的情况下,处理机 i 上的平均查询响应时间 $T(i)$ 近似为

$$T(i)=\mathrm{MAX}(T(i)_{cpu},T(i)_{disk},T(i)_{net}). \tag{1}$$

整个系统的查询响应时间为

$$T=\max(T_j,T_K)=\begin{cases}T(j) & \text{当 } Q\leqslant 1/P\\ T(k) & \text{当 } Q>1/P\end{cases}. \tag{2}$$

3　算法的代价公式

以往对于并行连接算法的研究模式是,提出一种改进的并行算法,然后对它进行性能分析[7].在这种模式中,代价分析处于被动地位,而且分析方法比较片面,对影响算法的因素考虑不全.在本文中,我们从一个新的角度出发,建立一种代价分析模型,用这种模型来评价并选择合适的并行连接算法.我们将综合考虑影响查询响应时间的诸多因素,建立一个基于多种硬件参数的代价分析模型,定性地推导 3 种 Hash 连接算法的代价公式.模型中描述了系统 3 个主要资源(CPU,I/O 和互联网)的重叠,刻画了数据偏斜(包括重分布偏斜和乘积偏斜)和同步延迟等因素对执行效率的影响.利用本文提出的代价模型,我们可以分析并行连接算法在各种硬件配置下的执行效率,为查询优化中并行连接算法的选取提供理论依据,对新的连接算法设计也具有一定的指导意义.

3.1　参数配置

P 表示处理机数;μ 表示处理机速度(cpu 每秒执行指令数);M 表示每个处理机的可用内存;R,S 表示参与连接的关系;β 表示连接选取率;γ 表示内存与小关系的比率;H 表示散列表膨胀系数;R_i,S_i 表示初始库划分后处理机 i 上的 R 和 S 片段;R'_i,S'_i 表示数据重新分配后处理机 i 上用来作连接的 Hash 桶;F_i 表示在有各种输出缓冲区的情况下,R'_i 的 Hash 表可被装入内存的部分;T_{io} 表示磁盘传送时间(以页为单位);T_{net} 表示网络传送时间(以页为单位);I_d 表示启动一次 I/O 的路径长(指令数);I_{comm} 表示在网络上交换一次消息的路径长;I_{comp} 表示比较两个属性的路径长;I_{Hash} 表示计算属性值的散列值的路径长;I_{build} 表示构造一个连接输出元组的路径长;I_{move} 表示在内存中移动一个元组的路径长;$|X|$ 表示 X 中的页数;$\{X\}$ 表示 X 中的元组数.

3.2　并行 Hash 算法代价公式

(1) 并行 Hybrid-Hash 算法

第 1 阶段

I/O 存取页面数:$N^1_{io}=(|R_i|+|S_i|)+(1-F_i)(|R'_i|+|S'_i|)$　(上标表示阶段,下同)

I/O 存取时间:$T(i)^1_{disk}=N^1_{io}*T_{io}$

从网络发送和接受的页面数:$N^1_{net}=(|R_i|+|S_i|)*(p-1)/p+(|R'_i|+|S'_i|)$

网络传送时间:$T(i)^1_{net}=N^1_{net}*T_{net}$

CPU 处理元组时间:

$$T(i)^1_{cpu}=(\{R_i\}+\{S_i\}+\{R'_i\}+\{S'_i\})*(I_{Hash}+I_{move})/\mu+N^1_{io}*I_d/\mu+N^1_{net}*I_{comm}/\mu$$

说明:为方便计算,将连接运算的时间统一在第 2 阶段来计算.

第 2 阶段

I/O 存取页面数:$N^2_{io}=(1-F_i)(|R'_i|+|S'_i|)$

I/O 处理元组时间:$T(i)^2_{disk}=N^2_{io}*T_{io}$

CPU 处理元组时间:

$$T(i)^2_{cpu}=(1-F_i)*(\{R'_i\}+\{S'_i\})*(I_{Hash}+I_{move})/\mu+\{S'_i\}*H*I_{comp}/\mu+$$
$$\{R'_i\}*\{S'_i\}*\beta*I_{build}/\mu+N^2_{io}*I_d/\mu$$

因而,最终处理机上的 I/O 存取时间:$T(i)_{disk}=((|R_i|+|S_i|)+2*(1-F_i)(|R'_i|+|S'_i|))*T_{io}$

网络传送时间:$N(i)_{net}=((|R_i|-|S_i|)*(p-1)/p)*T_{net}$

CPU 处理元组时间:

$$T(i)_{cpu}=(\{R_i\}+\{S_i\}+(2-F_i)(\{R'_i\}+\{S'_i\}))*(I_{Hash}+I_{move})/\mu+\{S'_i\}*H*I_{comp}/\mu+$$
$$\{R'_i\}*\{S'_i\}*\beta*I_{build}/\mu+((|R_i|+|S_i|)+2*(1-F_i)(|R'_i|+|S'_i|))*I_d/\mu+$$
$$((|R_i|+|S_i|)*(p-1)/p+(|R'_i|+|S'_i|))*I_{comm}/\mu$$

(2) Grace-Hash 算法

处理机 i 上的 I/O 存取时间:$T(i)_{disk}=((|R_i|+|S_i|)+2*(|R'_i|+|S'_i|))*T_{io}$

网络传送时间:$T(i)_{net}=((|R_i|+|S_i|)*(p-1)/p)*T_{net}$

CPU 处理元组时间:

$$T(i)_{cpu}=(\{R_i\}+\{S_i\}+2*(\{R'_i\}+\{S'_i\}))*(I_{Hash}+I_{move})/\mu+\{S'_i\}*H*I_{comp}/\mu+$$
$$\{R'_i\}*\{S'_i\}*\beta*I_{build}/\mu+((|R_i|+|S_i|)+2*(|R'_i|+|S'_i|))*I_d/\mu+$$
$$((|R_i|+|S_i|)*(p-1)/p+(|R'_i|+|S'_i|))*I_{comm}/\mu$$

(3) 并行 Simple 算法

处理机 i 上的 I/O 存取时间:$T(i)_{disk}=((|R_i|+|S_i|)+(1/\gamma-1)*(|R'_i|+|S'_i|))*T_{io}$

网络传送时间:$T(i)_{net}=((|R_i|+|S_i|)*(p-1)/p)*T_{net}$

CPU 处理元组时间:

$$T(i)_{cpu}=(\{R_i\}+\{S_i\}+(1/2\gamma+0.5)*(\{R'_i\}+\{S'_i\}))*(I_{Hash}+I_{move})/\mu+$$
$$(1/2\gamma+0.5)*\{S'_i\}*H*I_{comp}/\mu+\{R'_i\}*\{S'\}*\beta*I_{build}/\mu+$$
$$((|R_i|+|S_i|)+(1/\gamma-1)*(|R'_i|+|S'_i|))*I_d/\mu+$$
$$((|R_i|+|S_i|)*(p-1)/p+(|R'_i|+|S'_i|))*I_{comm}/\mu$$

4 结 论

在这一节中,我们将定性地分析上节推导出的公式,比较 3 种连接算法在不同主存容量、不同处理机个数、不同网络传输率和数据偏斜时的执行效率,从中得到一组有意义的结论.我们以自行研制的"并行数据库查询系统"为支撑[6],在以多台微机连成的局域网上进行了大量模拟实验.用 Wisconsin benchmark 基准测试得到的并行连接算法性能曲线,证明本文中所建立的代价分析模型是合理的.运用这个模型,可以方便地寻找系统资源并行度的平衡点,发现影响算法性能的瓶颈之所在,因而这种模型和分析方法为评价和选取并行算法提供了一条可行的途径.

在 Wisconsin benchmark 基准测试中连接两个关系 R 和 S,R 含有一万条元组(每条元组有 200Bytes),S 含有 10 万条元组(每条元组有 200Bytes).

4.1 主存容量对算法响应时间的影响

(1) Hybrid-Hash-Join 算法始终有较高的执行效率.

(2) Simple-Hash-Join 和 Hybrid-Hash-Join 算法都是对内存敏感的算法.当内存与小关系的比率 γ 从 0 到 1 变化时,算法的执行时间都会减少,当小关系可全部置入内存时,两个算法都有相同的执行时间.当 γ 从 1 降至 0.5 时,在 Hybrid-Hash-Join 算法中意味着有一半的关系要写回磁盘.当 γ 降至 0.5 以下时,Simple-Hash-Join 算法需将相同记录重复写回磁盘,效率急剧降低.

(3) Grace-Hash-Join 算法对内存不敏感.当内存减少时,连接阶段划分的桶数 B 增大,而桶数增加意味着增加一个小的调度开销.在 γ 降至 0.3 以下后,Grace-Hash-Join 与 Hybrid-Hash-Join 算法应有近似相同的响应时间.

(4) 运行 Hybrid-Hash-Join 算法要求每个处理机的最小内存量为 $2|R'_i|$,低于这个值,算法不能启动.以后随着内存的增加,执行时间降低.当内存增至与 R'_i 一样大时,执行时间明显降低.此后,再增大内存,执行时间的变换趋于平缓.这一结论意味着,只要处理机内存满足最低启动点加上一些缓冲开销,可将多余内存满足其他应用,也不致过分损害算法的性能.

图 1 刻画了 γ 在从 0 到 1 变化时,3 种算法的响应时间.

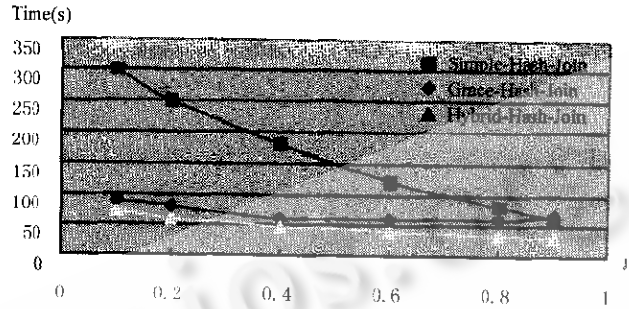


Fig. 1　The comparison of three algorithms' response time

图 1　3 种算法的响应时间比较

4.2　互连网传输率对算法响应时间的影响

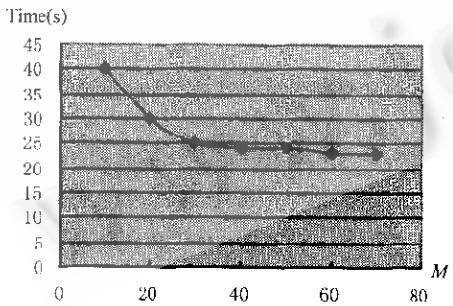
从代价公式可以看出,3 种算法的网络传输时间相同.因而在互连网传输率变化时,性能曲线有近似相同的变化趋势.

(1) 在高性能互连网中,网络带宽随处理机规模接近线性增加.因而,只有在网络传输率较低时,算法表现为网络受限,执行时间随传输率的增大而减小.在一个特定点以后,算法成为 CPU 受限,增大传输率对执行时间没有影响.

(2) 对于总线式互连网,网络带宽在所有处理机之间共享,且不随处理机数目增加而增大.当处理机数目很大时,算法将始终表现为网络受限,执行时间随着传输率的增大而线性降低,远远高于同样条件下高性能互连网的执行时间.

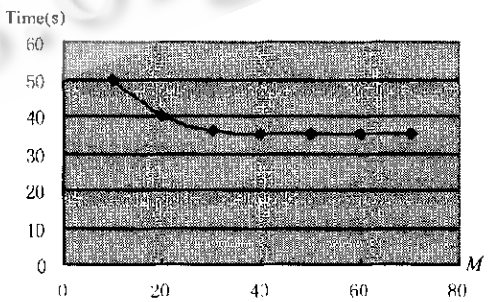
(3) 基于总线式互连网结构的并行系统用于并行连接查询的效率很低.

图 2(a)刻画了在高性能互连网中当网络传输率增加时,Hybrid-Hash-Join 算法响应时间的变化趋势.图 2(b)刻画了在单总线结构中当网络传输率增加时,Hybrid-Hash-Join 算法响应时间的变化趋势.



(a) High performance inter-network

(a) 高性能互连网络结构



(b) Single-Bus network

(b) 单总线结构的网络

Fig. 2　The response time at different network transfer rate in the parallel Hybrid-Hash-Join algorithm

图 2　并行 Hybrid-Hash-Join 算法网络传输率与响应时间的关系

4.3　处理机数对算法响应时间的影响

(1) 在高性能互连网络中,当处理机数较小时,算法是网络受限.增大处理机数执行时间会减少.以后随着处理机数的增加,算法成为 CPU 受限的,处理机数增加意味着增加系统的并行处理能力,算法的执行时间应有

线性加速.

(2) 在 SN 结构中,在具有大量处理机的并行系统中,关系全分段策略并不是适当的方法.文献[2]中定义最优并行度 n_0 为使一个操作的并行执行具有最小响应时间的处理机数,最优加速比 s_0 为一个操作以最优并行度并行执行时的加速比.令 N 表示关系中元组数目,C 表示处理一个元组所需的时间,S 表示调度器启动一个进程花费的时间,则 $(n_0)^2=CN/S$,$s_0=n_0/2$.可以看到,最优并行度与系统参数有关,而最大加速比只能是这个数目的一半.

图 3 刻画了在处理机数增加时 Hybrid-Hash-Join 算法响应时间的变化趋势.

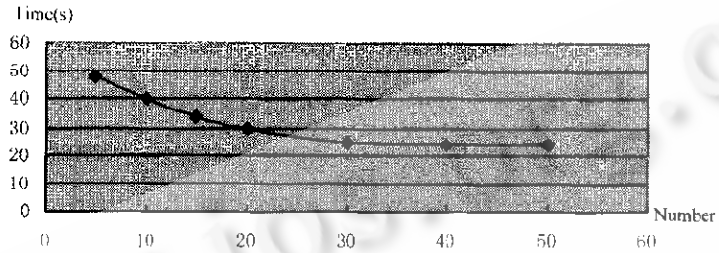


Fig. 3 The response time in various number of processors in the parallel Hybrid-Hash-Join algorithm

图 3 并行 Hybrid-Hash-Join 算法处理机数与响应时间的关系

4.4 I/O 与 CPU 并行度的影响

从上节的代价公式可知,I/O 与 CPU 并行度之间存在一个平衡点.在一定范围内算法是 I/O 受限的,增加每个处理机内的磁盘数,即增加磁盘的并行度可显著减少算法响应时间.这一结论告诉我们,分层并行结构对并行连接操作是可行的结构.

4.5 数据偏斜对算法响应时间的影响

(1) 数据偏斜对算法效率有严重影响.算法的执行时间总是由偏斜最严重的那个处理机的执行时间决定.

(2) 在分配阶段后,如果一个处理机得到大量元组,则可能发生桶溢出现象.因而,如果内存较大使得关系无需分桶,可减少数据偏斜带来的影响.

参考文献

1 DeWitt D J. A multiprocessor organization for supporting relational database management systems. IEEE Transactions on Computers, 1985,28(6):330~343

2 Biron D. Parallel algorithms for the execution of relational database operation. ACM Transactions on Database Systems, 1983,8(3):113~133

3 Yang Li. Parallel execution for complex query and implementation techniques for parallel database systems [Ph. D. Thesis]. Changsha: National University of Defense Technology, 1995
(杨利.复杂查询并行执行及并行数据库系统实现技术的研究[博士学位论文].长沙:国防科技大学,1995)

4 Schneider D A, Dewitt. A performance evaluation of four parallel join algorithms in a shared-nothing multiprocessor environment. In: Maier D ed. Proceedings of the ACM SIGMOD'89. Baltimore: ACM Press, 1989. 110~121

5 Chen M S, Yu P S. Scheduling and processor allocation for parallel execution of multi-join queries. In: IEEE Computer Society ed. Proceedings of the 8th International Conference on Data Engineering. Tempe, Arizona, Los Alamitos: IEEE Computer Society Press, 1992. 58~67

6 Walton B. A taxnomy and performance model of data skew effects in parallel joins. In: Lohman G M ed. Proceedings of the Very Large Data Base'91. CA: Morgan Kaufmann Publishers, Inc., 1991. 537~548

7 Seetha S Lakshmi, Philip S Yu. Effectiveness of parallel joins. IEEE Transactions on Knowledge and Data Engineering, 1990,2(4):410~424

8 Li Qing-hua, Gao Yan, Deng Chong *et al*. Research on parallel query system in database on MIMD model. High Technology Letters, 1995,5(6):1~5

(李庆华,高燕,邓冲等.MIMD模型上的数据库并行查询系统.高技术通讯,1995,5(6):1~5)

Analysis of the Efficiency of Parallel Hash-Join Algorithms Based on Shared-Nothing

LI Qing-hua　SUI Hai-yan　DENG Chong

(National High Performance Computing Center　Wuhan　430074)

(School of Computer Science and Technology　Huazhong University of Science and Technology　Wuhan　430074)

Abstract　Based on the shared_nothing construction, the various factors which may affect the query responding time of parallel join algorithms are analyzed. A cost analysis model is proposed to calculate the average task execution time on each processor and the total query responding time. The execution efficiencies of different parallel Hash algorithms based on the variation of memory are compared. The presented model and the analysis method have provided a feasible way to evaluate and choose parallel join algorithms.

Key words　Query parallel processing, parallel join, query responding time, model of performance evaluation.