排序合并 Join 算法的新结果*

孙文隽　李建中

（黑龙江大学信息技术研究所　哈尔滨　150080）

摘要　Join 操作是数据库中最昂贵和最常用的操作. 排序合并 Join 算法是实现 Join 操作的重要算法，得到了普遍接受并广为应用. 在重新研究了排序合并 Join 算法后发现，同时对两个 Join 关系进行外排序是不必要的，会带来很大的额外开销. 针对这个问题，提出了一种基于单关系外排序的分治 Join 算法，并在该算法的基础上提出了基于单关系外排序的并行分治 Join 算法. 理论和实验结果证明，基于单关系排序的分治 Join 算法高于排序合并 Join 算法. 特别是在并行计算环境下，基于单关系排序的并行分治 Join 算法的效率远远高于排序合并 Join 算法的并行版本.

关键词　Join 操作，Join 算法，排序合并 Join 算法.

中图法分类号　TP311

Join 操作是数据库系统中最昂贵最常用的操作. 在数据库操作算法的研究中，人们一直十分注重 Join 算法的研究，提出了大量算法[1~6]. 文献[2,6]对顺序和并行 Join 算法进行了全面系统的综述. 人们一般认为，Hash-Join 算法的效率最高，其次是 Sort-Meger-Join 算法，最后是 Nested-Loop-Join 算法. 事实上，各类 Join 算法在不同情况下的优劣不同. Hash-Join 算法要求选择高质量的 Hash 函数，实现起来较困难. Nested-Loop-Join 算法和 Sort-Merge-Join 算法思想简单，容易实现. 从效率和实现两方面考虑，Sort-Merge-Join 算法在商品化数据库系统中使用最多.

Sort-Merge-Join 算法分为 3 个阶段. 第 1 阶段对第 1 个 Join 关系进行外排序；第 2 阶段对第 2 个 Join 关系进行外排序；第 3 阶段合并两个已经排序的关系，产生 Join 结果. 本文对 Sort-Merge-Join 算法进行了重新研究，发现该算法对两个 Join 关系都进行外排序是没有必要的，这会引起很大的额外开销. 如果仅对小关系进行外排序，然后使用分治方法进行两个关系的 Join 操作，能节省大量时间. 基于这种思想，本文提出了一种新的基于单关系排序的分治 Join 算法（简称 SDC-Join 算法）. SDC-Join 算法首先对小关系进行完全排序；然后把已排序关系按内存大小划分成一组子集合；其次根据小关系划分情况对大关系进行划分，同时对各页进行内排序；最后进行分组合并，产生 Join 结果. SDC-Join 算法避免了大关系的外排序，节省了大量时间. 本文还对 SDC-Join 算法进行了并行化，提出了并行 SDC-Join 算法. 并行 SDC-Join 算法简称为 PSDC-Join 算法. 理论和实验结果证明，SDC-Join 算法的效率高于 Sort-Merge-Join 算法. 特别是在并行计算环境下，PSDC-Join 算法的效率远远高于 Sort-Merge-Join 算法的并行版本.

下面列出本文经常使用的符号和参数：

R，S：Join 关系；　　N_R：关系 R 的磁盘页数；

N_S：关系 S 的磁盘页数；　　T_R：关系 R 的元组数；

T_S：关系 S 的元组数；　　B_P：每个磁盘页的字节数；

T_P：每个磁盘或主存储器页的元组数；　T_{io}：一页磁盘读写所需要的时间；

$M+1$：可用主存储器空间页数（一个主存储器页的容量与一个磁盘页容量相等）；

* 本文研究得到国家杰出青年基金和黑龙江省杰出青年基金资助. 作者孙文隽，女，1965 年生，副教授，主要研究领域为数据库. 李建中，1950 年生，教授，博士生导师，主要研究领域为数据库.

本文通讯联系人：孙文隽，哈尔滨 150080，黑龙江大学信息技术研究所

本文 1998-01-06 收到原稿，1998-03-31 收到修改稿

T_{comm}:平均传输一个字节数据所需要的时间.

1 预备知识

首先我们来讨论顺序 Sort-Merge-Join 算法及其复杂性.顺序 Sort-Merge-Join 算法定义如下:

算法 Sort-Merge-Join.

输入:关系 R 和 S(R 的 Join 属性为 A,S 的 Join 属性为 B).

输出:R 和 S 在连接属性 A 和 B 上的连接结果 Result.

方法:

(1) 按照属性 A 的值排序关系 R;

(2) 按照属性 B 的值排序关系 S;

(3) FOR R 的每个元组 r,S 的每个元组 s DO

IF (r 在属性 A 上的值)=(s 在属性 B 上的值) THEN Result:=Result$\cup\{rs\}$.

由于 I/O 时间是顺序 Join 算法的主要时间开销.本文把磁盘存取时间定义为顺序 Join 算法的复杂性测度.设 Sort-Merge-Join 算法的输入主存储器缓冲区容量为 M 页,输出主存储器缓冲区容量为一页.Sort-Merge-Join 算法需要的磁盘存取时间为 $\mathrm{Cost(Sort\text{-}Merge\text{-}Join)}=T_{io}[2(N_R\log_M(N_R)+N_S\log_M(N_S))+(N_R+N_S+U)]$,其中 $2(N_R\log_M(N_R)+N_S\log_M(N_S))$ 为第(1)、(2)步进行 M 路合并排序所需要存取的磁盘页数,(N_R+N_S+U) 是第(3)步需要读的磁盘页数,U 是需要写的 Join 结果的磁盘页数.

现在我们来讨论并行 Sort-Merge-Join 算法及其复杂性.设 P 是并行计算机系统的处理机个数.并行 Sort-Merge-Join 算法简记为 PSM-Join,定义如下:

算法 PSM-Join.

输入:关系 R 和 S(R 的 Join 属性为 A,S 的 Join 属性为 B).

输出:R 和 S 在连接属性 A 和 B 上的连接结果 Result.

方法:

(1) 使用定义在属性 A 上的 Hash 函数 H 把 R 划分为 P 个子集合 $R_1,\ldots,R_P$,R_i 送处理机 $i(1\leqslant i\leqslant P)$;

(2) 使用定义 Hash 函数 H 按照属性 B 的值把 S 划分为 P 个子集合 $S_1,\ldots,S_P$,S_i 送处理机 $i(1\leqslant i\leqslant P)$;

(3) FOR $i=1$ TO P DO (并行地)

处理机 i 排序 R_i 和 S_i;

(4) FOR $i=1$ TO P DO (并行地)

结点 i 使用合并算法完成 R_i 和 S_i 的连接.

由于 I/O 时间和通信时间是并行 Join 算法的主要时间开销.我们定义并行 Join 算法的时间复杂性为磁盘存取时间和通信时间之和.算法 PSM-Join 的第(1)、(2)步的数据划分如下实现:把子集合 $\{r\in R\mid H(r[A])=j\}$ 和 $\{s\in S\mid H(s[A])=j\}$ 分配并传送到处理机 j.设 R 和 S 初始地存储在一个处理机上,第(1)、(2)步的并行磁盘页存取数为 $N_R+N_S+N_{Rmax}+N_{Smax}$,其中 N_{Rmax} 是 R 的 k 个子集合中最大子集合的磁盘页数,N_{Smax} 是 S 的 k 个子集合中最大子集合的磁盘页数.第(3)和第(4)步需要的并行磁盘页存取数与 Sort-Merge-Join 算法类似,不超过 $2(N_{Rmax}\log_M(N_{Rmax})+N_{Smax}\log_M(N_{Smax}))+(N_{Rmax}-N_{Smax}+U)$,其中 U 是各处理机 Join 结果中最大者的磁盘页数.于是,PSM-Join 算法需要的磁盘存取时间至多为 $\mathrm{Time(I/O,PSM\text{-}Join)}=T_{io}[N_R+N_S+2N_{Rmax}+2N_{Smax}+2(N_{Rmax}\log_M(N_{Rmax})+N_{Smax}\log_M(N_{Smax}))+U]$.PSM-Join 算法的通信操作只发生在第(1)和第(2)步,通信时间 Time(COMM,PSM-Join)不大于 $T_{comm}B_P(\frac{P-1}{P}N_R+\frac{P-1}{P}N_S)$,不小于 $T_{comm}B_P(P-1)(N_{Rmax}+N_{Smax})$.综上所述,PSM-Join 算法的时间复杂性为 Cost(PSM-Join)=Time(I/O,PSM-Join)+Time(COMM,PSM-Join).

2 SDC-Join 算法

定义1. 设 $R_1,\ldots,R_k$ 是 R 的 k 个子集合,$S_1,\ldots,S_k$ 是 S 的 k 个子集合,A 是关系 R 的 Join 属性,B 是 S 的

Join 属性.对于任意 R_i 和 $S_j(1\leqslant i,j\leqslant k)$,如果 R_i 在属性 A 上的投影与 S_j 在属性 B 上的投影不相交,则称(R_i,S_j)为 Join 无关集合对,否则,(R_i,S_j)称为 Join 相关集合对.

SDC-Join 算法是一种只需对小关系进行外排序的分治算法.在下边的讨论中,不失一般性,我们假定 R 小于 S. SDC-Join 算法分为如下3个阶段:

第1阶段(排序划分 R 阶段):按照 Join 属性值大小对关系 R 进行外排序,同时把关系 R 的每 M 个连续页作为一个子集合,R 划分为 $k=\left\lceil\frac{N_R}{M}\right\rceil$ 个子集合,第 i 个子集合记作 $SET_R[i]$,并计算出每个子集合 $SET_R[i]$ 在 Join 属性上的投影集合中的最小值 $\min_i$ 和最大值 $\max_i$.

第2阶段(划分 S 阶段):把关系 S 划分为 k 个子集合:对关系 S 的任一元组 t,如果 $\min_i\leqslant t$ 的 Join 属性值$\leqslant\max_i$,则 t 送第 i 个子集合.关系 S 的第 i 个子集合记作 $SET_S[i]$.显然,$(SET_R[i],SET_S[i])$是 Join 相关集合对,$(SET_R[i],SET_S[j])$ $(j\neq i)$是 Join 无关集合对.

第3阶段(Join 阶段):从1到 k 对关系 R 和 S 的 Join 相关集合对进行 Join 处理,并且合并成最终 Join 结果.

SDC-Join 算法的详细定义如下:

算法 SDC-Join.

输入:R,S:Join 关系;

输出:R 和 S 的 Join 结果 Result.

方法:

/* 第1阶段:排序划分 R 阶段 */

(1) 对 R 进行外排序,输出排序结果时把 R 划分为 $k=\left\lceil\frac{N_R}{M}\right\rceil$ 个子集合,每个子集合不超过 M 页,并计算每个子集合 $SET_R[i]$ 的 $\max_i$ 值和 $\min_i$ 值$(1\leqslant i\leqslant k)$;

/* 第2阶段:划分 S 阶段 */

(2) FOR S 中的每个元组 t DO

IF 存在 j 使得 t 的 Join 属性值在 $\max_j$ 和 $\min_j$ $(1\leqslant j\leqslant k)$之间 THEN t 分配到 $SET_S[j]$;

/* 第3阶段:Join 阶段 */

(3) FOR $i=1$ TO k DO

读 $SET_R[i]$ 进内存;

WHILE $SET_S[i]$ 未读完 DO

读下一页 $SET_S[i]$ 进内存,内排序该页,与内存中 $SET_R[i]$ 进行合并 Join,结果并入 Result.

下面我们分析 SDC-Join 算法的复杂性,从理论上与 Sort-Merge-Join 算法进行复杂性比较.我们先来分析 SDC-Join 算法需要的磁盘存取时间.SDC-Join 算法第1阶段与 Sort-Merge-Join 算法第(1)步基本相同,只是在最后输出排序结果时把 R 划分为 $\left\lceil\frac{N_R}{M}\right\rceil$ 子集合,同时计算每个子集合 $SET_R[i]$ 的 $\max_i$ 和 $\min_i$ 值,没有增加磁盘页存取数.所以,第1阶段需要存取的磁盘页数为 $2(N_R\log_M(N_R))$.第2阶段需要读写 S 一遍,存取的磁盘页数为 $2N_S$.第3阶段需要读一遍 R 和 S,写一次 Join 结果,磁盘存取页数为 (N_R+N_S+U),其中 U 是 Join 结果页数.综上所述,SDC-Join 算法需要的磁盘存取时间为 $\text{Cost(SDC-Join)}=T_{io}[2(N_R\log_M(N_R)+2N_S+(N_R+N_S+U)]$.

结论1. 如果 $N_S>M$,则 SDC-Join 算法的复杂性小于 Sort-Merge-Join 算法复杂性.

证明:由于 $\text{Cost(Sort-Merge-Join)}-\text{Cost(SDC-Join)}=T_{io}[2N_S\log_M(N_S))-2N_S]$,所以当 $N_S>M$ 时,$\text{Cost(Sort-Merge-Join)}>\text{Cost(SDC-Join)}$. □

一般情况下都有 $N_S>M$ 成立,所以,可以说一般情况下 SDC-Join 算法的性能高于 Sort-Merge-Join 算法.

3 并行 SDC-Join 算法

以下将并行 SDC-Join 算法简称为 PSDC-Join 算法.设并行计算系统具有 P 个处理机,每个处理机的内存可以容纳 $M+1$ 个磁盘页.PSDC-Join 算法分为如下4个阶段:第1阶段,使用 Hash 方法把关系 R 划分成 P 个子集

合,分布到P个处理机上;第2阶段,使用Hash方法把关系S划分成P个子集合,分布到P个处理机上;第3阶段,P个处理机分别使用SDC-Join算法,完成Join操作.算法PSDC-Join的详细定义如下:

PSDC-Join算法.

输入:R和S是Join关系,A是R的Join属性,B是S的Join属性.

输出:R和S的Join结果Result.

方法:

/* 第1阶段:分布R */

(1) 在属性A上使用Hash函数H把R划分为P个子集合$R_1,\ldots,R_P$,R_i送处理机i($1\leqslant i\leqslant P$);

/* 第2阶段:分布S */

(2) 在属性B上使用Hash函数H把S划分为P个子集合$S_1,\ldots,S_P$,S_i送处理机i($1\leqslant i\leqslant P$);

/* 第3阶段:并行Join */

(3) FOR $i=1$ TO P DO(并行地)

处理机i使用SDC-Join算法完成R_i和S_i的Join.

下面分析PSDC-Join算法的复杂性,并从理论上与PSM-Join算法进行复杂性比较.PSDC-Join算法第1阶段和第2阶段与算法PSM-Join第(1)和第(2)步相同.设关系R和S初始存储在一个处理机上,这两个阶段的并行磁盘页存取数为$N_R+N_S+N_{Rmax}+N_{Smax}$,其中$N_{Rmax}$是R的$k$个子集合中最大子集合的磁盘页数,$N_{Smax}$是S的$k$个子集合中最大子集合的磁盘页数.由SDC-Join算法的磁盘页存取数的分析可知,第3阶段的并行磁盘存取页数为$2N_{Rmax}\log_M(N_{Rmax})+2N_{Smax}+(N_{Rmax}+N_{Smax}+U)$,其中$U$是各处理机Join结果中最大者的磁盘页数.于是,PSDC-Join算法需要的并行磁盘存取时间为 $Time(I/O,PSDC\text{-}Join)=T_{io}[N_R+N_S+2N_{Rmax}+4N_{Smax}+2N_{Rmax}\log M(N_{Rmax})+U]$. PSDC-Join算法的通信时间复杂性Time(COMM,PSDC-Join)与PSM-Join算法相同,即不大于$T_{comm}B_P\left(\frac{P-1}{P}N_R+\frac{P-1}{P}N_S\right)$,不小于$T_{comm}B_P(P-1)(N_{Rmax}+N_{Smax})$.综上所述,PSDC-Join算法的时间复杂性为Cost(PSDC-Join)=Time(I/O,PSDC-Join)+Time(COMM,PSDC-Join).

结论2. 如果$N_{Smax}>M$,则SDC-Join算法的复杂性小于Sort-Merge-Join算法复杂性.

证明:由Cost(PSM-Join)−Cost(PSDC-Join)$=T_{io}[2N_{Smax}\log_M(N_{Smax})-2N_{Smax}]$可知,当$N_{Smax}>M$时,Cost(Sort-Merge-Join)>Cost(SDC-Join). □

一般情况下都有$N_{Smax}>M$成立,所以,可以说一般情况下SDC-Join算法的性能高于Sort-Merge-Join算法.

4 实验结果

为了比较SDC-Join算法和Sort-Merge-Join算法的性能,我们使用PC/586微型计算机在UNIX环境下实现了这两个算法.在实验中,我们设主存储器每页容量为1 024字节,每个磁盘页的容量为1 024字节,系统可用的主存储器为6页,其中5页作为输入(4页用于小关系,1页用于大关系),1页用于输出Join结果.以下用R表示小关系,用S表示大关系.我们首先为R和S随机地产生了16对关系实例,然后在每对关系实例上分别执行SDC-Join算法和Sort-Merge-Join算法.表1给出了实验结果.表中第1栏和第2栏的数据量是关系中存储的整型数.表中的第3栏和第4栏给出了SDC-Join算法和Sort-Merge-Join算法的运行时间,运行时间以1/60s为度量单位.从表1可以看出,SDC-Join算法的效率高于Sort-Merge-Join算法.

为了比较PSDC-Join算法和并行Sort-Merge-Join算法的性能,我们在计算机机群并行环境下实现了这两个算法.计算机机群由5台PC/586通过以太网连接而成,通信协议是TCP/IP,操作系统是UNIX.5台计算机中1台机器作为前置机,4台机器作为后端机.后端机负责并行Join.前置机协调后端机并行运行.Join关系的产生方法同上.表2给出了实验结果.表中的数据量是指关系中的整型数.运行时间以1/60s为度量单位.表2说明,PSDC-Join算法的效率高于并行Sort-Merge-Join算法.

表1 SDC-Join 算法和 Sort-Merge-Join 算法运行时间的比较

关系 R 数据量	关系 S 数据量	SDC-Join 算法	SORT-MERGE-Join 算法
2 048	2 048	18	32
2 048	4 096	19	42
2 048	8 192	20	66
2 048	16 384	22	113
4 096	1 024	27	36
4 096	2 048	28	41
4 096	4 096	37	75
4 096	8 192	44	103
4 096	16 384	47	152
8 192	1 024	67	85
8 192	2 048	68	91
8 192	4 096	69	106
8 192	8 192	71	132
8 192	16 384	73	171
16 384	8 192	96	154
16 384	16 384	116	320

表2 PSDC-Join 算法和 PSM-Join 算法的并行运行时间比较

关系 R 数据量	关系 S 数据量	PSDC-Join 算法	并行 Sort-Merge-Join 算法
4 096	1 024	27	36
4 096	2 048	28	41
4 096	4 096	18	28
4 096	8 192	23	37
4 096	16 384	30	53
8 192	4 096	24	34
8 192	8 192	31	45
8 192	16 384	38	61
16 384	8 192	47	60
16 384	16 384	133	346

参考文献

1 Lu H, Ooi B C, Tan K L. Query Processing in Parallel Relational Database Systems. Los Alamitos, CA: IEEE Computer Society Press. 1994. 147～157

2 Mishra P, Eich M H. Join processing in relational databases. ACM Computing Surveys, 1992,24(1):63～113

3 DeWitt D *et al*. Implementation techniques for main memory database systems. In: Bestrice Y ed. Proceedings of ACM-SIGMOD International Conference'84 on Management of Data. New York: Association for Computing Machinery, 1984. 1～8

4 Schneider D A, DeWitt D. A performance evaluation of four parallel Join algorithms in a shared-nothing multiprocessor environment. In: Proceedings of ACM-SIGMOD International Conference'89 on Management of Data. New York: Association for Computing Machinery, 1989. 110～121

5 Bitton D. Parallel algorithms for the execution of relational database operations. ACM Transactions on Database Systems, 1983,8(3):324～353

6 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)

Sort-Merge-Join Algorithm Revisited

SUN Wen-jun LI Jian-zhong

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

Abstract The Sort-Merge-Join algorithm is an effective and widely used algorithm for implementing the important Join operation in database systems. The algorithm is revisited in this paper. It is discovered that sorting both operand relations externally is not necessary in the algorithm. The cost of the algorithm would be reduced greatly if only one operand relation is sorted externally. In order to overcome the shortcomings of the Sort-Merge-Join algorithm, a new Join algorithm called SDC-Join algorithm, is proposed in this paper. The SDC-Join algorithm is a single-relation-sorting based divide-and-conquer algorithm. A parallel version of the SDC-Join algorithm is also presented in the paper. Theoretical analysis and experiment results show that the performance of the SDC-Join algorithm is much higher than that of the Sort-Merge-Join algorithm in both uniprocessor computer systems and parallel computer systems.

Key words Join operation, Join algorithm, Sort-Merge-Join algorithm.