JOIN 并行数据库 负载平衡 关系划分 粒度 (17)

75-77

计算机科学 1995Vol. 22№. 1

基于负载平衡的并行 JOIN 关系划分粒度研究*)

葛芝宾 谢 立 陈 军 金志权
(南京大学计算机科学与技术系 南京 210008)

TP311.13

摘 要 Based on the analysis of the task sizes and the load, this paper discusses the granularity of relation spliting in the spliting phase taking acount of task load being less than average load, and probes into the relationship between the granularity and load balancing. The minimum number of buckets is determined on the basis of relation spliting granularity, and the maximum number of product tuples of each node is induced under the prerequisite for ensuring load balance in parallel join.

关键词 Parallel join, Data skew, Parallel join load balancing, Spliting granularity.

1. 引言

在并行数据库系统中，关系的并行 JOIN 操作，须首先将二个关系按 JOIN 属性的取值进行对应的划分，每个关系被划分成若干个桶(Bucket)，二个关系的对应桶组成一个个桶对，即一个个 JOIN 任务，以便分派给系统中若干个结点并行地进行 JOIN 操作。然而，可能的数据扭曲(Data Skew)，以及 JOIN 负载不平衡，成为影响并行 JOIN 性能的突出问题，就一般而言，数据是引起负载不平衡的主要原因，而数据扭曲的特殊情形，便是在 JOIN 属性上取某一相同值的元组数目过多，因此，有效地处理数据扭曲便成为解决并行 JOIN 负载平衡的关键之一。从最近的研究来看，虽然提出了一些可对数据扭曲进行处理的方法[1,2]，但从实验数据可以看出，处理效果尚未达到应有的水平，负载平衡仍存在明显差距。究其原因，是因为其算法中的重要参数的确定缺少相应的理论根据，从而未能适当地确定这些参数。例如，文[2]中的虚处理机数目根据什么确定，尚未给出明确的答案。实质上，这些参数与本文讨论的问题有关，简言之，就是当参与 JOIN 的关系中在 JOIN 属性上取某一相同值的元组很多时，关系划分的粒度，即任务划分为多大才能保证不存在超过平均负载的任务。本文从分析不同的 JOIN 任务的不同负载量入手，以(从静态的角度)保证负载平衡为目标，分析讨论关系划分的最大粒度问题，并得出相应的结论；同时据此还推出一些有用的结论。

2. 负载平衡与任务大小的关系

如前所述，在对关系 R、S 并行 JOIN 之前，须按 JOIN 属性取值进行划分，设 R、S 被分别划分为 m_1、m_2 个桶，即 $R=\bigcup_{i=1}^{m_1}R_i$，$S=\bigcup_{j=1}^{m_2}S_j$，则 $R\bowtie S$ 的任务 T 可表示为：$T=\{T_{i,j}\mid i=1,\cdots m_1; j=1,\cdots m_2\}$，这里，我们考虑到在对数据扭曲进行处理时，有可能须将某一关系的部分元组多次复制的情况，即 i 与 j、m_1 与 m_2 有时可能并不对应相等。为了叙述的方便，我们采用如下的符号：$|R|$——关系 R 的元组数；L_R——关系 R 的元组长度(字节数)。对不同对象使用上述符号表示类似意义。

定义 1 设 $R\bowtie S$，其 JOIN 结果生成率 $P_r=|R\bowtie S|/(|R|\cdot|S|)$。

显然，对任意进行 JOIN 的关系 R 和 S，有 $0\leqslant P_r\leqslant 1$。

定义 2 设 $R_i\bowtie S_j$ 是并行 JOIN 的一个任务 $T_{i,j}$，则任务 $T_{i,j}$ 的大小 $S_{T_{i,j}}$ 为：$S_{T_{i,j}}=|R_i|\cdot|S_j|$。其中：$i=1,\cdots m_1$；$j=1,\cdots m_2$。

在对关系进行划分之后，我们可能得到各种大小的任务，而各个任务的 JOIN 负载量不仅与任务大小有关，而且与任务的结果生成率有关。很显然，若某个任务的负载量超过平均负载，则必然导致并行 JOIN 时的负载不平衡，这种情况的出现通常由数据扭曲引起，其特殊情形是由 JOIN 属性上对应地取某个(些)值的元组组成的任务太大。要保证任何一个任务不超过平均负载的负载量，关键是对上述情形的任务的处理。

结论 1 设 $R\bowtie S$，$P_r\leqslant 1/N$，不超过平均负载的且纯是 JOIN 属性上取某一值的重复元组的任务大小应不大于：$(|R|\cdot|S|/N)\cdot P_r(1+C)/(1+N\cdot P_r\cdot C)$，其中，N 为系统中的结点数，C 为一参数

*)江苏省自然科学基金资助项目。葛芝宾 江苏省盐城师专讲师，南京大学计算机系访问学者。

(详见下文)。

证明:考虑这样二个任务:$T_{1,1}=R_1\bowtie S_1$ 和 $T_{2,2}=R_2\bowtie S_2$,使它们满足如下条件:

(1)$R_2\bowtie S_2$ 为最理想的平均负载,即有:$|R_2|=|R|/N$,$|S_2|=|S|/N$;且它生成的结果元组总数为平均数$(|R|\cdot|S|\cdot P_r|)/N$,即该任务的结果生成率为$|R_2\bowtie S_2|/(|R_2|\cdot|S_2|)=N\cdot P_r$。

(2)$|R_1|=|R_2|$,$|S_1|=|S_2|$,$R_1\bowtie S_1$ 的结果生成率为1。

(3)执行这二任务 JOIN 操作的结点的性能和采用的算法完全相同。

那末,在 JOIN 阶段,仅为完成这二任务 JOIN 操作的开销(不计入子桶收集的开销)可分为如下二个主要部份:

(1)从外存将任务中的元组读入内存的 IO 开销,CPU 为检查元组匹配所作的开销。对于上述二任务,这部份开销二者完全相同,我们记为 R_C。

(2)将匹配的元组进行合并,输出结果元组至外存,我们用符号$(R_i\bowtie S_i)_{M-w}$表示任务 $T_{i,i}$的这部份开销。由于二任务的结果生成率不同,JOIN 的结果元组数目不同,因而二任务的这部份开销成如下关系:

$$(R_1\bowtie S_1)_{M-w}/(R_2\bowtie S_2)_{M-w}=1/(N\cdot P_r) \qquad (1)$$

从而,在 JOIN 阶段二任务的工作量之比 t 应为

$$t=\frac{R_C+(R_1\bowtie S_1)_{M-w}}{R_C+(R_2\bowtie S_2)_{M-w}} \qquad (2)$$

变形为:

$$\frac{(R_1\bowtie S_1)_{M-w}}{(R_2\bowtie S_2)_{M-w}}=t+\frac{R_C}{(R_2\bowtie S_2)_{M-w}}(t-1)$$

令 $C=\frac{R_C}{(R_2\bowtie S_2)_{M-w}}$,并将(1)式代入,有

$$\frac{1}{N\cdot P_r}=t+C\cdot(t-1)$$

解之得

$$t=\frac{C\cdot N\cdot P_r+1}{N\cdot P_r\cdot(1+C)} \qquad (3)$$

(3)式说明,任务 $T_{1,1}$的 JOIN 工作量,是任务 $T_{2,2}$的 t 倍,而要达到并行 JOIN 时的负载平衡,也就要使得最大负载的时间开销不超过理想平均负载,这就意味着应将任务 $T_{1,1}$的尺寸缩小 t 倍,因而

$$S_{T1,1}=\frac{|R|}{N}\cdot\frac{|S|}{N}\cdot\frac{N\cdot P_r\cdot(1+C)}{1+N\cdot P_r\cdot C}$$

$$=\frac{|R|\cdot|S|}{N}\cdot\frac{P_r\cdot(1+C)}{1+N\cdot P_r\cdot C} \qquad (4)$$

从而结论 1 成立。

在(4)式中,参数 C 代表结点处理机为 JOIN 应读取的平均元组数所花的 IO 开销与生成平均结果元组的开销之比。它与系统性能、关系 R 和 S 的元组数及元组长度,以及结果生成率 P_r 和 JOIN 操作的算法有关。在推导过程中,我们以理想的平均任务作为模板,因此,(4)式在$|R_i|$、$|S_i|$的元组数符合$|R|$与$|S|$的比例关系时是精确的,而如果符合(4)式的任务中,$|R_i|$、$|S_i|$并不满足$|R|$、$|S|$之间的比例,并且 R、S 的元组长度偏差甚大,则将产生读取元组的 IO 代价的偏差,但从(4)式可以看出这一误差对(4)式的任务大小的影响较小,必要时可作适当修正。

以上我们未对 $P_r=0$ 和 $P_r>1/N$ 的情形进行讨论,首先因为这二种情形很少发生;其次因为,当 $P_r=0$ 时,不存在满足结论 1 条件的任务,而当 $P_r>1/N$ 时,容易证明所有结点必须完成的任务大小都大于$|R|\cdot|S|/N^2$,我们通常不会将任务划分得这样大。

3. 几个相关的结论

当 R 和 S 的 JOIN 结果生成率 $P_r<1/N$ 时,完全由 JOIN 属性上的重复元组组成的任务,与相同尺寸的其它任务相比,其 JOIN 负载最重。当这样的任务的尺寸大于(4)式时,我们须对其按(4)式大小进行细分(Subdivide),所有这样的任务细分后,尺寸等于(4)式的任务总数满足如下结论。

结论 2 当 $0<P_r<1/N$ 时,将所有由 JOIN 属性上重复元组组成的任务按(4)式进行细分,得到的尺寸恰好等于(4)式的任务总数少于结点数 N。

证明:(用反证法)设那样的任务总数$\geq N$,不妨设为 N 个,则由(4)式,这样的 N 个任务产生的结果元组总数为

$$N\cdot\frac{|R|\cdot|S|}{N}\cdot\frac{P_r\cdot(1+C)}{1+N\cdot P_r\cdot C}$$

$$=|R|\cdot|S|\cdot\frac{P_r\cdot(1+C)}{1+N\cdot P_r\cdot C}$$

那么,仅以这 N 个任务的结果元组数计算 R⋈S 的结果生成率,按定义 1 应至少为

$$\frac{P_r\cdot(1+C)}{1+N\cdot P_r\cdot C}>P_r \qquad (P_r<1/N\text{ 时})$$

显然与已知矛盾。故结论 2 成立。

对于某些并行 JOIN 算法(如 ABJ$^+$算法[3]等),任务是在分桶阶段划分完成的,而在分桶阶段一般又不易知道每个任务的结果生成率,当 R、S 的结果生成率较低时,(4)式限定的任务也较小,这就可能使得一些包含符合结论 1 条件、但大于其任务大小的任务不被察觉并加以处理,这就需要对关系划分的最大任务进行限定,这就是通常所说的关系划分粒度问题。当关系划分粒度满足(4)式时,若能将由 JOIN 属性上的重复元组组成且其尺寸恰好等于(4)式的任务首先分派出去,则结论 2 说明,至少有一个结点必须执行二个这样的任务的情况不会发生。也就是说,这样的关系划分为并行 JOIN 时各结点的负载趋于平衡创造了可能的条件。

另一方面,当 JOIN 采用类似于 ABJ$^+$算法时,

在分桶阶段关系 R 和 S 被分别划分成 m 个桶，为了能得到希望的粒度，m 的最小取值可据如下结论确定。

结论 3 欲使分桶阶段的关系划分粒度满足结论 1 的任务大小，则分桶数 m 应满足

$$m \geqslant \sqrt{\frac{N \cdot (1+N \cdot P_r \cdot C)}{P_r \cdot (1+C)}}$$

证明：设分桶后 R、S 的 m 个桶大小（元组数）为关于 m 的平均值，则由结论 1 应有

$$\frac{|R|}{m} \cdot \frac{|S|}{m} \leqslant \frac{|R| \cdot |S|}{N} \cdot \frac{P_r \cdot (1+C)}{1+N \cdot P_r \cdot C}$$

于是有

$$m^2 \geqslant \frac{N \cdot (1+N \cdot P_r \cdot C)}{P_r \cdot (1+C)}$$

从而得证。

假定 CPU 的速度远远大于 IO 速度，即在 JOIN 阶段对结果元组的处理时间仅为结果元组转储的 IO 开销，那末，

$$(R_i \bowtie S_j)_{M-W} = P_r \cdot N \cdot |R_i| \cdot |S_j| \cdot (L_{Ri}+L_{Sj}) = \frac{P_r \cdot |R| \cdot |S|}{N} \cdot (L_R+L_S) \quad (5)$$

代入(4)式右端的第二个因式，有

$$\frac{P_r \cdot (1+C)}{1+N \cdot P_r \cdot C} = \frac{P_r + \frac{N \cdot R_C}{|R| \cdot |S| \cdot (L_R+L_S)}}{1+\frac{N^2 \cdot R_C}{|R| \cdot |S| \cdot (L_R+L_S)}}$$

对于某个具体应用，N、|R|、|S|、L_R、L_S、R_C 均是确定的，因此，上式是关于 P_r 的单调增函数；从而(4)式也是关于 P_r 的单调增函数。当 $P_r \leqslant 1/N$ 时，结论 1 给定的任务大小不大于 $|R| \cdot |S|/N^2$。另一方面，当任务中的元组完全是 JOIN 属性上重复的元组时，其 JOIN 产生的结果元组数正好等于任务大小，因此有：

结论 4 设 $R \bowtie S$，$P_r \leqslant 1/N$，在满足负载平衡的前提下，任何一个结点的 JOIN 结果元组总数不超过 $(|R| \cdot |S|)/N^2$。

结论 4 说明当 $0 < P_r \leqslant 1/N$ 时，在满足负载平衡的情形下任一结点产生的 JOIN 结果元组数存在上界，这一结论，也告诉我们应为存储 JOIN 结果预置多少外存空间。

4. 关于参数 P_r、C

前面推导的结论中，用到参数 P_r 和 C，并已说明了它们的意义，现在我们对这二个参数的计算和获取作一简短的讨论。

前已述及，参数 C 与 |R|、|S|、L_R、L_S、P_r 和所采用的 JOIN 算法有关，现假设采用类似于 ZIGZAG[1] 的算法对任务进行 JOIN，且 $|R| \geqslant |S|$，JOIN 时的内存缓冲区大小(s-1 Blocks)将 |R|/N 个元组的数据分为 K 份；设 ω_{io} 为系统的 IO 速率，CPU 检查元组是否匹配的速率为 ω_{cpu} 那么

$$R_C = \frac{|R| \cdot L_R + |S| \cdot L_S \cdot K}{N \cdot \omega_{io}} \perp \frac{|R| \cdot |S|}{N^2 \cdot \omega_{cpu}}$$

再将式(5)除以 ω_{io} 代入 C 式，即可求得 C。

对于结果生成率 P_r，其精确值一般只有在 JOIN 运算以后才能得出，但其近似值是可以获取的，可分如下二种情况：(1)在对某两个关系或其上的某二属性初次 JOIN 之前，可通过采用类似于文[4]的抽样方法进行抽样，利用定义 1 计算出 P_r 的近似值作为其初始值。(2)对于已做过 JOIN 运算的情形，我们必有一些已知的 P_r 值、或直接利用前一次 JOIN 得到的 P_r 值；或者将以前的若干 P_r 值利用预测公式进行预测计算。有关公式或方法请参见相应文献。显然，情形(1)对不同的关系或属性只出现一次，而对于一些实用的数据库，情形(2)则是经常发生的；也就是说，参数 P_r 在绝大多数情况下是可知的，也是较接近准确值的。

5. 结束语

本文的分析讨论，从一个侧面揭示了并行 JOIN 的数据扭曲与负载平衡之间的关系，并对任务尺寸、结果生成率和负载量的内在关系给出了量化描述，相关地推出了几个实用的结论。本文的结论中虽然采用了某些参数，但避免了以某个确定算法为基础，使得具有普遍意义。同时，本文给出的结论并非纯理论的研究，也是实用的。当我们清楚地了解了我们所必须认识的客观存在，以及这些客观存在间的内在联系，就有可能对一些已有的算法进行改进和完善，使其发挥更好的性能；同时，我们也可能进一步研究和发现更新更好的算法或策略。

参考文献

[1] Joel L. Wolf et al., A Parallel Sort Merge Join Algorithm for Managing Data Skew. IEEE Transactions on Parallel and Distributed System, Vol. 4, No. 1 1993

[2] David J. Dewitt et al., Practical Skew Handling in Parallel Joins. Proc. of the 18th VLDB Conf., 1992

[3] Kien A. Hua and Chiang Lee, Handling Data Skew in Multiprocessor Database. Computers Using Partition Tuning. Proc. of the 17th Intl. Conf. on VLDB, 1991

[4] Richard J. Lipton et al., Practical Selectivity Estimation through Adaptive Sampling. In Proc. ACM SIGMOD 1990