第 24 卷第 6 期　　四川理工学院学报(自然科学版)　　Vol. 24　No. 6
2011 年 12 月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Dec. 2011
文章编号:1673-1549(2011)06-0677-05

基于启发函数的网格数据库查询算法

金圣华[1],朱浩杰[1],李　伟[2]

(1.淮阴工学院计算机工程学院,江苏　淮安　223001;2.南京航空航天大学计算机科学与技术学院,南京　210016)

摘　要:查询处理技术是网格数据库研究的核心,但是传统的查询处理技术无法适应动态变化的网格环境,导致网格资源利用率低、查询效率不高。给出了基于副本的网格数据库查询处理代价模型和两种查询处理算法,利用不同的数据集和网格环境参数进行实验验证。实验结果表明,基于启发函数的网格数据库查询算法在一定程度上提高了查询处理效率,生成子查询时间代价为 O(n),消耗时间比采用穷举策略下降 10% 左右。

关键词:网络数据库;查询处理;启发函数
中图分类号:TP311　　文献标识码:A

引　言

网格数据库[1]是数据库技术和网格技术相结合的一门新的研究内容。主要是对网络上大量的数据资源进行有效管理,从而为用户提供统一的访问接口。网格数据库系统主要是用于接收用户提交的查询,对查询进行解析和优化,将优化后的查询发送到网格节点上并行执行,最后将最终的查询结果返回给用户。

数据库副本技术主要包含副本创建与迁移、副本的一致性维护以及基于副本的查询处理技术[2-3]。目前,在网格数据库系统中,基于单缓存的查询处理技术效率不高,稳定性也比较差,不适用于网格数据库环境。本文首先给出了基于缓存的查询处理的代价模型;然后在此基础上设计了一个启发函数,考虑节点负载能力,提出了基于启发函数的网格数据库查询处理算法;最后,在不同的数据集和网格环境参数下,对算法的性能进行了比较验证。实验得出基于启发函数的查询处理算法充分利用网格节点资源,提高了查询处理效率。

1　网格数据库查询处理算法

1.1　网格数据库查询处理框架

网格数据库查询处理框架主要由三部分组成:用户接口层、中间协调层和分布式查询执行层。首先,用户接口层是接收用户提交的查询,将查询提交至中间协调层进行处理。中间协调层接收到用户的查询请求后,其查询编译解释器根据元数据信息对查询进行解析和改写,形成系统内部的统一表示形式,然后由查询协调器结合元数据信息、代价模型以及查询分解算法将查询任务分解为多个子查询;子查询发送到相应的网格节点去执行。分布式查询执行层是并行执行子查询。最后,系统再对子查询反馈回来的查询结果进行合并,向用户接口层提交最终的查询结果。

1.2　代价模型

在网格数据库系统查询处理过程中,用户查询 Q 被分解生成 n 个子查询并发送到不同的网格节点上执行;子查询结果返回给查询协调器并进行合并,最终合并后的查询结果返回给用户。其中,查询的时间消耗分为四个部分:查询解析时间 Tparse、查询拆解时间 Tsplit、子查询执行和数据传输时间、数据合并时间 Tmerge,由于子查询并行执行,因此子查询的时间消耗为 n 个子查询中时间消耗最大值,则查询 Q 代价消耗可以表示为:

$$Cost(Q) = T_{pase} + T_{split} + Max(cost(Qi)) + T_{merge},\quad i \in [1,n] \tag{1}$$

由于查询 Q 的解析、拆解及数据合并在查询协调器中完成,相对于子查询的执行和传输的时间消耗可以忽略。因此,查询处理的主要的时间消耗在子查询的执行

收稿日期:2011-10-25
作者简介:金圣华(1979-),女,江苏淮安人,讲师,硕士,主要从事生物信息学中的机器学习及人工智能方面的研究,(E-mail)52594205@qq.com

和数据传输时间,那么查询 Q 的代价近似可以表示为:

$$Cost(Q) \approx Max\{cost(Qi)\}, i \in [1,n] \quad (2)$$

根据公式(2)可知:提高系统的查询处理效率关键在于如何利用各个节点数据、计算及通信资源对查询 Q 进行合理拆分。

1.3 查询拆分策略

网格数据库系统接收到用户查询 Q 后,如图 1 所示,首先根据元数据筛选出能够被当前查询使用的副本;接着,根据副本中的条件范围对查询 Q 中的 RP 进行拆分,并根据元数据信息、网格节点通信和计算能力从而形成多个子查询;最后,将子查询发送到对应的节点上去执行并返回最终结果。

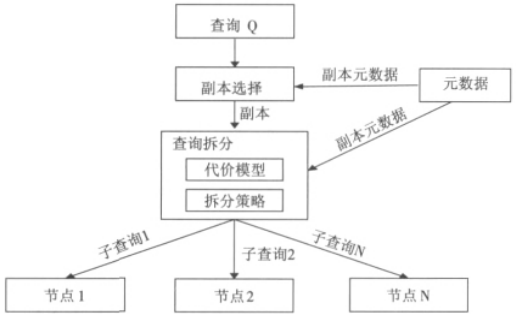

图 1 查询拆解过程

1.3.1 查询分解算法

用户查询 Q 提交给系统。首先选择副本,得到副本集合 RS;接着利用 RS 副本中范围对 RP 进行拆解;RP 被划分为 m 个子区域($rp_1, rp_2, \cdots, rp_m$);每个子区域对应的数据被副本中数据覆盖,分为两种情况:(1)多个副本覆盖其数据;(2)仅被一个网格节点数据覆盖。

问题划归为采取何种策略利用副本对查询 Q 中范围进行最优拆分,其中网格数据包含矩阵 $S_{n,m}$:n 为网格节点数目,m 为查询 Q 中条件 RP 被副本中条件 RP 划分的数目,其中矩阵行对应网格节点,列对应子区域。如果节点 i 上不包含区域 j 对应数据,那么 $S_{i,j}$ 取 0;反之 $S_{i,j}$ = Size(DataRP(RP_j)),Size(DataRP(RP_j)) 为区域 RP_j 对应的数据大小。解矩阵 $X_{m,n}$:m 为子区域数目,n 为网格节点数目,$X_{i,j}$ 的取值为 0 或 1,表示区域 rp_i 对应数据是否在第 i 节点上获得,若 $S_{i,j}=0$,则 $X_{i,j}=0$。由于子区域对应的数据获取一次,因此 X 的每一行中有且仅有一个 1。访问代价数组 B [n]:B [i] 为查询协调器节点和其他节点之间的带宽。矩阵进行相乘得到代价矩阵 C:

$$S \times X \times B^{-1} = C$$

C 是一个 N* 1 的矩阵,C 中每一行数据:

$$C1 = (S_{1,1} * X_{1,1} + S_{1,2} * X_{2,1} + \cdots + S_{1,m} * X_{m,1}) / B_1$$

$$C2 = (S_{2,1} * X_{1,2} + S_{2,2} * X_{2,2} + \cdots + S_{2,m} * X_{m,2}) / B_2$$

$$\cdots\cdots$$

$$Cn = (S_{n,1} * X_{1,n} + S_{n,2} * X_{2,n} + \cdots + S_{n,m} * X_{m,n}) / B_n$$

求解 X,使得取到 Min{Max(Ci)},i [1,n],由于网格系统中节点数和区域被划分数目都不确定,根据文献[4-5],设计了基于穷举策略和基于启发函数两种算法。

算法 1:(穷举法)

算法思想:列出 X 所有的可能的解,求出每一种的代价消耗,最终选择代价最小的解 X。

算法描述:

输入:查询 Q

输出:查询结果 Result

1. 提取出 FS,TS,RP,选择出符合条件的副本集合 RS,其中 FS 为查询字段集合,TS 为关系表集合,RP 为范围查询条件,;

2. 利用 RS 中每个副本对应的 rp 对 Q 中 RP 区域进行分解,得子区域 $RP_1, RP_2, \cdots, RP_n$;

3. 初始化矩阵 S、X,tempX,minCost;

4. WHILE (nextXState (X))

5. curcost = coutCost (X,S,B);

6. IF (minCost > curcost)

7. bestX = X; minCost = curcost;

8. END IF

9. END WHILE

10. SQ = generateSubQuery (bestX,S);

11. Result = executeAndMerge (SQ);

12. RETURN Result;

算法 2:(启发式算法):

算法:Cost(RP_i,S_j) = Size(Data(RP_i)) /B_j + Cost(S_j),Cost(RP_i,S_j) 为区域 RP_i 对应数据在节点 S_j 上获取时,S_j 节点总的代价,Cost(S_j) 为网格节点 S_j 上任务队列中已有查询时间消耗总和。

算法思想:

利用副本的条件 RP 将查询 Q 的 RP 拆解为多个子区域,通过元数据得到子区域对应数据被副本覆盖情况。然后,对子区域进行排序,排序策略分为两种:(1)根据子区域对应数据被副本覆盖数从小到大排序,副本越少越优先调度;(2)根据子区域对应数据量从大到小排序,数据量越多越优先调度。根据元数据信息生成子查询,根据启发函数计算出每个子查询在副本节点上总代价,选择代价最小的网格节点作为执行当前子查询的节

点。

算法描述:

输入: 查询 Q

输出: 查询结果 Result

1. 取出查询 Q 中 Fs,Ts,RP,筛选副本,得到符合条件的副本元数据集合 RS;

2. 利用 RS 中条件对 RP 进行拆分,拆分出子区域 RP_1,RP_2,…,RP_m;

3. RPS = SORT(RP_1,RP_2,…,RP_m) ;

4. 初始化矩阵 B、S,节点负载数组 Cost ;

5. FOR EACH RP_j(RP ∈ RPS)

6. minCost = MAX_VALUE, bestIndex = -1;

7. FOR i = 1 TO n

8. IF S [i] [j] ! =0

9. IF minCost > Size(Data(Rp_j)) /B [i] + Cost(j)

10. minCost = Size(Data(Rp_j)) /B [i] + Cost(j) ;

11. bestIndex = i;

12. END IF

13. END IF

14. END FOR

15. Cost [bestIndex] + = minCost;

16. SQ [i] = generateSQL(RP_j, best 节点 Index) ;

17. END FOR

18. Result = ExecuteAndMerge(SQ) ;

19. RETURN Result;

通过对算法的分析知: 采用穷举法算法生成子查询的时间消耗为 O(nn) ,而采用启发式算法生成子查询的时间消耗为 O(n) 。

2　实验分析

2.1　实验环境

20 台 PC 机组成的网格数据库环境,操作系统为: 15 台采用 Windows XP 操作系统、5 台采用 Linux 操作系统,软件为: 自主开发的网格数据库原型系统 NH-GridDB。数据集采用 Benchmark TPC - D 数据库测试数据集。数据随机分布在网格各个节点上。采用网格模拟器 Optorsim 提供的动态带宽数据来模拟各节点之间的带宽变化,Optorsim 提供的带宽数据采集自现实网络。图 2 给出了带宽在一天 24 小时之内的变化情况,其中带宽均值为真实值较带宽峰值的比例。图 3 则给出了峰值为 100kb/s 情况下带宽在一天 24 小时内变化的一个实例。

2.2　实验结果及分析

对于算法 1 进行下列实验,实验使用的节点数为

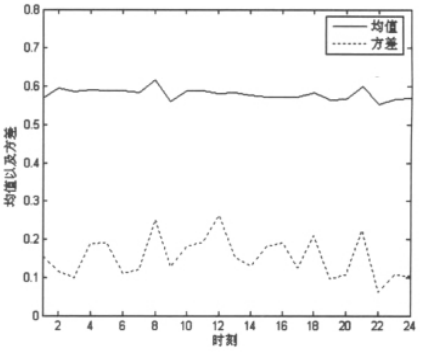

图 2　带宽的均值以及方差变化情况

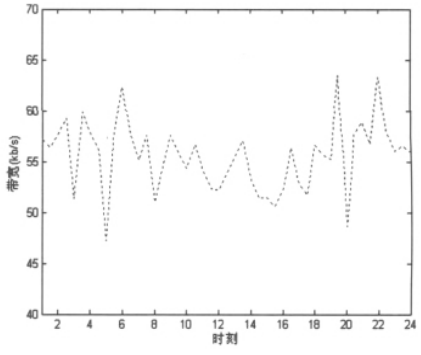

图 3　带宽变化情况

15,子区域数目依次从 1 递增到 10,查询区域为 20000。实验结果如下:

图 4 为采用穷举算法,在网格环境其他参数不变的情况下,横坐标为节点数目,纵坐标为生成子查询的时间(毫秒) 。图 4 表明随着子区域数目线性增加,穷举法生成子查询的时间消耗呈指数级增长。图 5 表示采用穷举法查询时间消耗,可以得到当查询范围一定时,随着区域被拆解的越多,执行查询消耗的时间呈递减趋势。但是由于穷举法生成子查询时间消耗随着节点数目指数级增长,故当节点增长到一定程度时,穷举法生

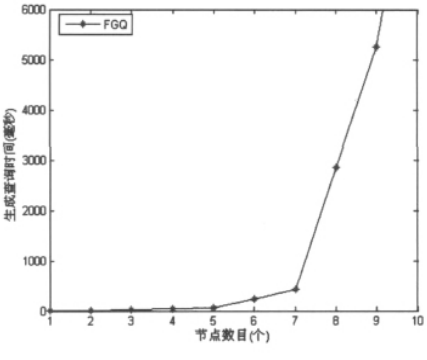

图 4　穷举法生产最终查询时间消耗

成子查询时间消耗太大,因而穷举算法在节点数目较多的情况下不可采用。

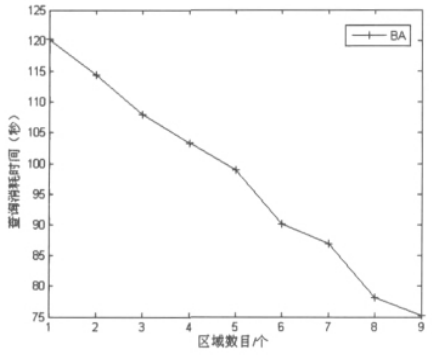

图5 穷举法查询时间消耗

对于启发式算法,设计了下列实验: 网格节点数 15 查询区域为 20000,子区域从 1 依次增长到 15。当网格数据库系统中节点数、查询范围不变,而查询范围被拆解数目变多时候,实验结果如下:

图 6 表明随着查询范围被拆解成的子区域变多,穷举法的生成子查询时间消耗指数级增长,而启发式算法的时间消耗则比较稳定; 图 7 表明随着查询范围被拆解的子区域增多,执行查询的时间消耗呈递减趋势,并且启发式算法与穷举法的时间消耗相差在 10% 左右。当网格数据库系统中节点数不变,被拆分的子区域数量不变时候,查询区域不断增大,实验结果如下:

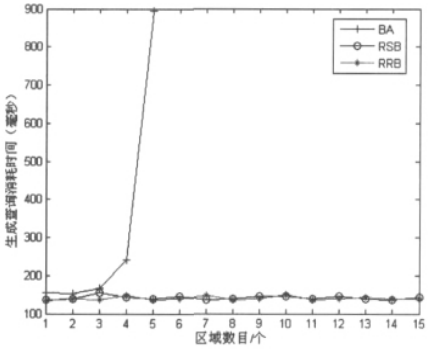

图6 生成查询消耗时间

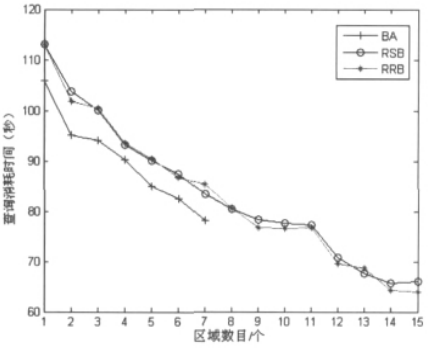

图7 执行查询消耗时间

图 8 表明随着查询范围的增加,穷举法生成子查询的时间远大于启发式算法。图 9 表明随着查询范围的不断增加,启发式算法中两种调度方法的时间消耗相差在 10% ~15% 左右,因此,可以采用启发式算法来进行查询范围的拆解。

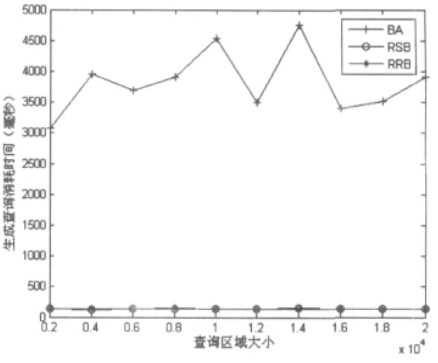

图8 生成查询消耗时间

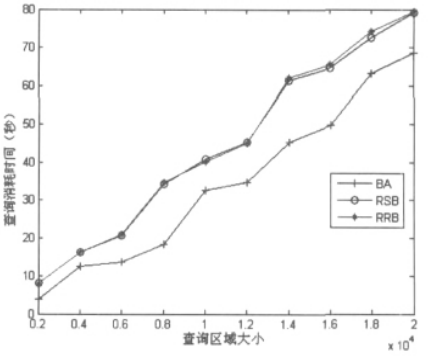

图9 执行查询消耗时间

3 结束语

本文将查询区域拆分为多个子区域; 根据网络带宽、启发函数等将每个子区域调度到最佳网络节点,生成对应的子查询,在不增加额外处理代价的情况下,实验结果证明利用启发式算法能够很好地利用副本中数据提高查询处理效率。

参 考 文 献:

[1] 王 珊,张坤龙. 网格环境下的数据库系统 [J]. 计算机应用,2004,24(10):1-4.

[2] Jeffrey F,Qiong Luo Naughton. Form-Based Proxy Caching for Database-Backed Web Sites [C]. Proceedings of the 27th VLDB Conference Roma,Italy,2001.

[3] Tanu Malik,Randal Burns,Amitabh Chaudhary. Bypass Caching. Making Scientific Databases Good Network Citizens [C]. ICDE 2005.

[4] Yu Min-Max. Optimization of Several ClassicalDiscrete Optimization Problems [D]. Plenum Publishing Corporation,1998.

[5] Luss H,Smith D R. Resource Allocation among Competing Activities [J]. A Lexicographic Min-Max Approach,Operations Research Letters,1986,5:227-231.

[6] Pang J S,Yu C S. A Minimax Resource Allocation Problem with Substitutions [J]. European Journal of Operational Research,1989,41:218-223.

[7] Luss H. A Nonlinear Minimax Allocation Problem with Multiple Knapsack, Constraints [J]. Operations Research Letters,1991,10:183-187.

[8] Klein R S,Luss H,Rothblum U G. Minimax Resource Allocation Problems with Resource Substitutions Represented by Graphs [J]. Operations Research,1993,41:959-971.

[9] Yu G,Kouvelis P. Complexity Results for a Class of Minimax Problems with Robust Optimization Applications,Complexity in Numerical Optimization [J]. World Scientific Publishing Company, Singapore, 1993, 98(1):501-509.

Heuristic Function Based Query Algorithm in Grid Database

JIN Sheng-hua[1], *ZHU Hao-jie*[1], *LI Wei*[2]

(1. Department of Computer Science and Technology, Huaiyin Institute of Technology, Huaian 223001, China;
2. Department of Computer Science and Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: As one of the key technologies of the network database, traditional query processing technology can not adapt to dynamic Network environment. It will result in lower utilization of Network resources and lower query efficiency. The query processing cost model is advanced, two corresponding algorithms are given and the experiments are carried out under different data sets and Network environment parameters. Experimental results show that the heuristic algorithm improves the efficiency of Network database query processing to a certain degree. The cost of generating sub queries is O(n), compared with the Exhaustive method, the query time decreases 10%.

Key words: network database; query processing; heuristic function