网格数据的语义查询优化研究

罗永红[1,2],陈特放[1],张友生[3]
LUO Yong-hong[1,2],CHEN Te-fang[1],ZHANG You-sheng[3]

1.中南大学 交通运输工程学院,长沙 410083
2.长沙理工大学 计算机与通信工程学院,长沙 410076
3.湖南师范大学 物理与信息科学学院,长沙 410081
1.School of Traffic and Transportation Engineering,Central South University,Changsha 410083,China
2.College of Computer and Communication Engineering,Changsha University of Science and Technology,Changsha 410076,China
3.College of Physics and Information Science,Hunan Normal University,Changsha 410081,China

LUO Yong-hong,CHEN Te-fang,ZHANG You-sheng.Study on semantic query optimization of grid data.Computer Engineering and Applications,2009,45(2):16-20.

Abstract: In the environment of heterogeneous data sources integration using grid technology,the problem of semantic query can be solved through introducing ontology.In order to improve the performance of semantic query in grid,the paper proposes a GSQO(Grid Semantic Query Optimizer),GSQO mainly implements optimization of the following three modules:(1)Semantic extension of user query;(2)Resource selection;(3)Parallel processing.The result of experiment shows that GSQO provides better query performance through using foregoing optimization strategies.

Key words: grid;ontology;semantic query;Grid Semantic Query Optimizer(GSQO)

摘 要:利用网格技术实现的异构数据源集成环境中,引入本体可以解决网格数据的语义查询问题。为了提高网格环境中语义查询的效率,提出了一个基于本体的语义查询优化器(GSQO),该优化器主要实现了以下3个模块的优化:(1)用户查询语义扩展;(2)资源选择;(3)并行处理。实验结果表明,GSQO通过采取上述优化策略提供了较好的查询效率。

关键词:网格;本体;语义查询;网格语义查询优化器
DOI:10.3778/j.issn.1002-8331.2009.02.005 **文章编号:**1002-8331(2009)02-0016-05 **文献标识码:**A **中图分类号:**TP393

本体[1]共享概念模型明确的形式化规范说明,本体的目标是获取、描述和表示相关领域的知识,提供对该领域知识的共同理解,确定该领域内共同认可的词汇(术语),并从不同层次的形式化模式上给出这些词汇和词汇之间相互关系的明确定义。在基于网格的数据库集成环境中,将本体技术运用到信息查询中,构造基于本体的语义查询较好地解决了传统信息查询中由于缺乏语义支持所造成的问题。由于查询中引入了本体,因此查询过程中不仅需要考虑异地异构的分布式查询处理问题,还需考虑本体的语义映射问题,这样显然增加了查询的复杂性。所以在基于网格的语义查询中,查询处理的优化显得更为重要。

网格中的优化工作通常是作为调度处理的一部分而实现的,在特定的时间内用户请求获得计算资源而得到执行。目前,具有优化功能的调度器主要有以下两种:(1)AppLeS(Application Level Scheduler)[2],是一种高效的调度器,能为各种网格应用提供优化的调度。AppLeS给每一个网格应用分配了一个AppLeS代理,这样就可以过滤掉低级的资源并生成一个优化的调度;(2)Condor[3]是一个高吞吐量系统,具有内置的中央调度模块。它利用一个代理去匹配任务的资源请求和来自于资源提供者的资源描述,一旦发现有匹配,代理就会将指定的资源分配给对应的任务。此外,其他的优化工作主要考虑的是基于网格的数据库系统,文献[4]提到的APG(Active Proxy-G service),是一个在网格环境中利用高速缓存来优化查询的中间件;文献[5]中提出的OGSA-DQP(Open Grid Architecture-Distributed Query Processor)利用两个网格服务来完成查询处理,分别是:GDQS(Grid Distributed Query Service)和GQES(Grid Query Evaluation Service),GDQS负责将查询计划分解为子查询计划并调配给不同的结点;在每一个结点GQES负责执行子查询计划并返回查询结果。

尽管前面所述的优化工作考虑了网格中的查询处理和优化问题,但是要促进查询效率的提高还需要进一步的研究。比如,OGSA-DQP主要作用是一个分布式查询处理器,而不是一

基金项目:国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2006AA11Z230);国家自然科学基金(the National Natural Science Foundation of China under Grant No.60674003);河南省教育厅科研基金项目(No.08C101)。
作者简介:罗永红(1975-),男,博士研究生,主要研究领域为网格计算、智能运输系统;陈特放(1959-),男,博士,教授,博士生导师,主要研究领域为计算机网络、机车故障智能诊断;张友生(1969-),男,博士后,副教授,主要研究领域为软件工程、移动计算及智能软件。
收稿日期:2008-09-15 **修回日期:**2008-10-14

个优化器;APG 仅采用高速缓存来进行优化;AppLeS 和 Condor 利用调度策略来计划网格应用的执行,但忽略了一些很重要的效率因素(比如数据传输)。

针对上述情况,本文提出了一个基于本体的网格语义查询优化器(GSQO),该优化器主要通过以下 3 个部分实现查询的优化:(1)用户查询语义扩展;(2)资源选择(3)并行处理。

1　GSQO 的体系结构

GSQO 主要由 3 个主模块(用户查询语义扩展、资源选择和并行处理)和 3 个辅助处理组成,见图 1。用户查询语义扩展模块主要采用基于 BGP(基本图模式)的 SPARQL 查询优化策略对本体进行查询,实现对用户查询的快速语义扩展;资源选择模块的输入为 AQT(抽象查询树见图 4,每一个叶子结点表示等值连接中的一个基本关系,每一个内部结点表示它所对应的叶子结点处理结果的操作),主要是采用一个融入了几个效率相关因数的排位函数,对于一个特定的操作按照有效资源所需花费的代价进行排位最后选择花费代价最小的资源执行该操作;并行处理模块通过计算归一化数据传输代价 NDTC[6](Normalized Data Transmission Cost,数据传输代价/边的通信代价)来实现查询计划的并行处理。为了提供查询优化运行时的信息,GSQO 创建了 3 个辅助服务[7]:EIS(环境信息服务)、DIS(数据信息服务)、TPS(传输预测服务)。EIS 负责提供网格环境中主机的静态和动态信息,比如负载、CPU 速度、有效 RAM 等。DIS 管理网格中基本关系的副本目录,并负责将副本目录返回给有请求的客户端。对于一个给定的关系,DIS 能提供维护该关系副本的主机和对该关系的统计(比如关系的大小,字段的长度等)。对于给定查询中的一个关系,TPS 负责评估候选主机和其他查询所涉及主机传输效率的比较。

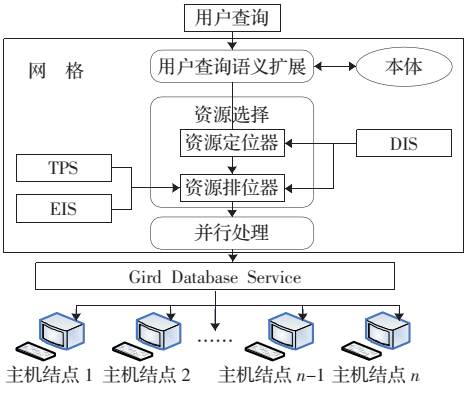

图 1　GSQO 的体系结构

2　用户查询语义扩展优化

在本文中,用户查询语义扩展优化主要是通过基于 BGP 的SPARQL[8]对本体查询进行优化,获取查询关键词的等价、父子以及兄弟等关系,从而实现高效的语义扩展。

2.1　BGP 的图论

BGP(Basic Graph Pattern)用 B 表示,本文将 B 定义为一个图 G[9],这个图由无向连通图集 U 组成,其中 $U=\{g_1,g_2,g_3,\cdots\}$。对于 U 中的每一对$(g_i,g_j)\in U$,g_i和 g_j 都是不连通的。每一个图 g 用一个序偶(N,E)表示,其中 N 为不同三元组模式组成的无序集(即 g 的结点集),E 为不同三元组模式对组成的无序集(即 g 的边集)。

一个三元组模式由主体、谓词和客体 3 个部件组成,一个三元组模式对至少共享一个绑定或非绑定的部件。在文中,把三元组模式对称之为连接的三元组模式。因此,具有共享变量的两个三元组模式和具有相同主体 URI 的两个三元组模式都是连接的,本文只考虑非绑定部件(即变量)所确定的连接。对于一个 $BGP(B)$,不连通的(g_i,g_j)对的执行顺序不影响查询效率,因此只需考虑每一个 $g\in U$ 内部的优化问题。

定义 1　$g\in U$ 中 N 的尺寸为 g 的结点数,即 $g\in U$ 中三元组模式的个数。

定义 2　p_g为 $g\in U$ 的一个执行计划,是已确定好的 g 的结点序列。

定义 3　P_g为 $g\in U$ 的执行计划空间,p_g为 P_g的元素,P_g所包含的元素数为 $g\in U$ 的执行计划总数。

假定 g 中结点数为 N,则 P_g所包含的元素数为 $N!$。因此,即使是一个只包含少量三元组模式的简单 BGP,它的执行计划空间也相当大。

一个 $BGP(B)$的执行计划 p_B为一个无序集 S,S 的元素为执行计划 p_g。S 所包含的元素数等于 U 所包含的元素数,也就是说每一个连通图 $g\in U$ 有一个关联的执行计划 p_g,从而 p_g为 S 的元素。一个执行计划 $p_g\in P_g$能用一个有向无环图(DAG)表示。定义 D_g为 P_g中的有向无环图集合。每一个 $d_g\in D_g$表示无向连通图 $g\in U$ 的一个或多个执行计划。图 3(a)所显示的 DAG 对应的是自顶向下执行一个 BGP 实例 $E1$(见图 2)的一个执行计划。对于任意两个结点$(i,j)\in d_g$,i 和 j 之间有一条路径,如果对应于 i 和 j 的三元组模式是连接的,并且 i 在查询计划中首先执行,则在 P_g和 D_g之间存在一个很明显的关系,可以用下列函数进行表示:$f:p_g\rightarrow d_g$,f 是单射函数,而不是满射函数。因此一个查询计划 p_g可以唯一地映射到一个 d_g,而一个 d_g可以提取出一个或多个 p_g。例如,按自顶向下顺序(即 1,2,3,4,5,6 的序列)执行图 2 的 BGP 三元组模式的执行计划唯一地映射到图 3(a)所显示的 DAG d_g,然而这个 DAG 也可以抽取出查询序列为(2,3,1,4,5,6)的三元组模式查询计划。因此,D_g的大小和 P_g的大小一般是不相等的。

```
?X  rdf:type  ub:GraduateStudent
?Y  rdf:type  ub:University
?Z  rdf:type  ub:Department
?X  ub:memberOf  ?Z
?Z  ub:subOrganizationOf  ?Y
?X  ub:undergraduateDegreeFrom  ?Y
```

图 2　一个 BGP 实例 $E1$

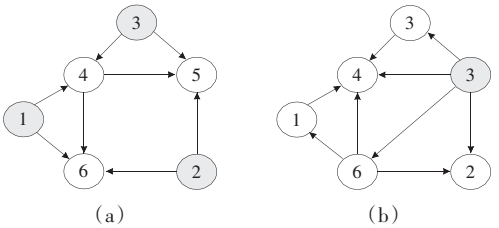

图 3　$E1$ 的 DAG 和优化后 $E1$ 的 DAG

2.2　三元组模式的选择性

一个模式的选择性和它与三元组的匹配概率有很紧密的

联系,选择性和概率的取值范围都为区间[0,1]。选择性模式,例如[ex:s ex:p ex:o],有一个值为0的下界,匹配三元组的概率很低;非选择性模式,例如[?s ?p ?o],有一个值为1的上界,匹配三元组的概率很高。

2.2.1 三元组模式的选择性

三元组模式的选择性本文采用下列公式进行计算:$sel(t)=sel(s)*sel(p)*sel(o)$,$sel(t)$表示三元组模式$t$的选择性,$sel(s)$为主体$s$的选择性,$sel(p)$为谓词$p$的选择性,$sel(o)$为客体$o$的选择性。选择性的值位于区间[0,1],对应于匹配一个模式的三元组数目和数据集中三元组总数的归一化。由于非绑定成分必须匹配数据集中的每一个三元组,因此非绑定三元组模式成分(即变量主体、谓词和客体)的选择性一般为1.0。

一个绑定主体的选择性采用下列公式计算:$sel(s)=1/R$,R表示数据资源的总数。一个绑定谓词的选择性本文采用下式进行计算:$sel(p)=T_p/T$,T_p为匹配谓词p的三元组数,T为数据集中三元组的总数。

对于一个绑定客体的选择性的评估,一般采用下列公式进行计算:

$$sel=\begin{cases} sel(p,o_c), \text{如果 } p \text{ 为绑定谓词} \\ \sum_{p_i \in P} sel(p_i,o_c), \text{其它} \end{cases}$$

在上式中,(p,o_c)对表示客体o落入谓词p所对应的三元组直方图[10]分组。$sel(p,o_c)=hc(p,o_c)/T_p$,表示落入(p,o_c)分组中的三元组数和匹配谓词p的三元组数的比值。如果谓词p为非绑定的,客体选择性的评估需考虑每一个谓词的直方图(即客体的选择性为每一个谓词p_i所对应的$sel(p_i,o_c)$的累加和)。

2.2.2 连接三元组模式的选择性

包含非绑定主体和客体的关联谓词的连接三元组模式的结果集具有一个上界,这个上界的大小为连接三元组模式的结果集大小。如果一个连接三元组模式P的上界为U_p,则P的选择性计算公式为:$sel(P)=U_p/T^2$,这里的T^2表示数据集中所有三元组模式的平方。对于绑定主体或客体的连接三元组模式的选择性计算需在上式的基础上添加一个因子,这个因子为一个三元组模式中绑定成分(主体和客体)的选择性函数。比如,P是一个连接三元组模式,包含三元组模式[?x p_1 o]和[?x p_2 ?y],则P的选择性可按下式进行计算:

$$sel(P)=U_p/T^2*sel(p_1,o_c)$$

其中$sel(p_1,o_c)$表示P中第一个三元组模式的客体选择性。

2.3 本体查询优化算法

基于BGP的SPARQL本体查询的核心优化算法的伪代码如下:

```
Begin
  Nodes(g)→N; //N 为 g 的结点集
  Edges(g)→E; //E 为 g 的边集
  EP={Ø}; //EP 为查询执行计划,初值为空
  SelEdgeMinSel(E)→e; //从 E 中选择具有最小选择性的边赋给 e
  OrderNodesBySel(e)→EP;  //对 e 的两个结点按选择性进行排序,然后依次加入 EP
  while size(EP)≤size(N) do
    SelEdgeMinSelVisitedNode(EP,E)→e; //从 E 中选择包含一个已访问结点的具有最小选择性的边(未访问过的边)赋给 e
    SelNotVisitedNode(EP,e)→EP; //选择 e 中未访问过的结点加入 EP
  end while;
  return EP
End;
```

该核心优化算法首先选择g中具有最小选择性的边,该边所对应的结点标记为已访问过的结点,并按照结点选择性从小到大的顺序依次加入到最后的查询执行计划p_g中。选择了第一条边之后,该算法就会反复选择满足下列两个条件的边:(1)具有最小的选择性;(2)包含一个已访问过的结点。因此,每一次循环就会有一个新的结点加入到执行计划EP中。

优化算法所生成的执行计划的有向无环图具有一个公共的特点:图中只有一个结点只包含流出的有向边(即执行计划中首先执行的结点)。只包含流出有向边的结点由于没有和该结点前面部分的执行计划合作,因而导致产生了两个中间结果集的笛卡儿积。例如,按自顶向下顺序执行图2的执行计划所提取的DAG见图3(a),首先执行的3个三元组模式(分别为1、2、3结点,它们都只包含流出有向边)建立了两个笛卡儿积。而图3(b)所示的优化执行计划永远也不会产生笛卡儿积,这是因为该图中只包含一个具有流出有向边的结点(即结点5),并且该结点首先被执行。

3 资源选择

3.1 平均数据传输等待时间指数—TLR

主机之间对数据的平均传输等待时间经常被用来衡量一台主机的传输能力,但存在这样一个问题:数据的分配严重影响了平均传输等待时间值,即或高或低的任意异常值能对平均传输等待时间产生很大影响。本文采用了一个指数—TLR(Transmission Latency Reputation)来解决上述问题,TLR的计算[7]如下:

假设涉及关系$R_1,R_2,R_3,\cdots,R_n$的查询Q在时间t内执行,关系R_i有M个副本,分别存储于主机$H_{i1},H_{i2},H_{i3},\cdots,H_{iM}$。在时间$t$内,存储了$R_i$的主机$H_{ij}$的TLR被计算为一个带权的平均值:

$$TLR_{ij}=\frac{\sum_{k=1}^{N}\sum_{l=1}^{M}\sigma(ij,kl)*TL(ij,kl)}{\sum_{k=1}^{N}\sum_{l=1}^{M}\sigma(ij,kl)}(k\neq i)$$

式中$TL(ij,kl)$为时间t内H_{ij}和H_{kl}之间的平均传输等待时间。如果H_{ij}和H_{kl}所存储的关系中只有关系R_i涉及查询Q,则TLR_{ij}的计算不应该包括H_{kl},这是因为在执行查询Q的时候H_{ij}和H_{kl}之间没有传输发生。$\sigma(ij,kl)$是分配给H_{ij}的权,它的值为$S^2(ij,kl)/\max_{ij}(S^2)$,其中$S^2(ij,kl)$为$H_{ij}$和$H_{kl}$之间的传输等待时间的变化值,$\max_{ij}(S^2)$为$H_{ij}$和其他主机之间的传输等待时间的最大变化值。将$\sigma(ij,kl)=S^2(ij,kl)/\max_{ij}(S^2)$代入$TLR_{ij}$的计算式得到:

$$TLR_{ij}=\frac{\sum_{k=1}^{N}\sum_{l=1}^{M}S^2(ij,kl)*TL(ij,kl)}{\sum_{k=1}^{N}\sum_{l=1}^{M}S^2(ij,kl)}(k\neq i)$$

通过引入 $\sigma(ij,kl)$,主机的平均传输等待时间就得到了调节。如果一台主机在数据传输时的平均等待时间异常值越多越极端,则 TLR 的值越大,这样它的传输能力和其他主机比较起来就显得越差。当选择资源提供者的时候,后面部分要讨论的排位函数使用 TLR 作为衡量一台主机传输能力的指数。

3.2 主机选择

在资源选择模块中,为了给 AQT 中的每一个基本关系发现合适的主机,需要子模块资源定位器反复地访问 AQT。对于每一个基本关系,资源定位器通过访问 DIS,然后返回一个存储了请求基本关系的候选主机列表,这个列表紧接着传给另外一个称之为资源排位器的模块,资源排位器通过与 DIS、TPS 和 EIS 联系从而获得所有候选主机的动态和静态的统计信息。对于存储了关系 R_i 的候选主机 H_{ij},排位器采用了一个基于代价函数的排位函数[11]来计算 H_{ij} 的排位,排位函数是 5 个带权的归一因数值的线性组合,其表达式如下:

$$rank_{ij}=((mips_{ij}-\min(mips))*\omega_{cpu})/(\max(mips)-\min(mips))+((ram_{ij}-\min(ram))*\omega_{ram})/(\max(ram)-\min(ram))+((count_{ij}-\min(count))*\omega_{count})/(\max(count)-\min(count))+\omega_{\omega_k}/((\omega_{k_{ij}}-\min(\omega_k))/((\max(\omega_k)-\min(\omega_k))+1))+\omega_{TLR}/((TLR_{ij}-\min(TLR))/(\max(TLR)-\min(TLR))+1))$$

其中,$mips$ 表示 H_{ij} 的 MIPS(每秒百万条指令),ω_{mips} 表示 MIPS 的权重,ram_{ij} 表示主机 H_{ij} 上的有效 RAM 容量,ω_{ram} 表示 RAM 的权重,$count_{ij}$ 表示主机 H_{ij} 所存储的涉及查询的关系数,ω_{count} 表示 $count$ 的权重,$\omega_{k_{ij}}$ 表示 H_{ij} 的当前负载(0 表示空闲,1 表示忙),ω_{ω_k} 表示负载的权重。

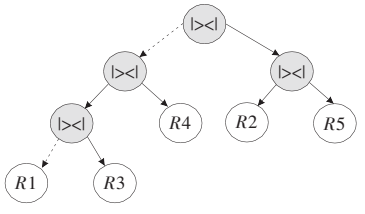

图 4　一个 AQT 的例子

在排位函数中引入 $count$,增加存储了多个关系的主机的获得较高排位的机会。一台主机的排位越高,越有可能执行的是本地连接处理。在查询优化处理实现中,所有的权重值都设置为 1;而在特殊的环境中,根据因数是起决定作用还是起次要作用,可以对相应的权重值进行调整。

对于存储了某个关系的所有候选主机的排位都计算完了之后,然后选择排位最高的主机作为该基本关系的提供者。由于排位函数中的各个因数会严重影响到查询的执行效率,因此对于一个给定查询中的某一基本关系,候选主机所获得的排位反映了其参与查询的适合性。一般来说,候选主机的排位越高,选择作为查询所涉及关系的提供者的机会就越大。

4 并行处理

4.1 数据传输代价

将关系从一个结点传输到另一个结点的代价称之为数据传输代价,影响数据传输代价的主要因素为真正的数据传输量和传输通信代价。数学公式描述如下:

$$DTC_{ij}=D_{ij}*CC_{ij}$$

其中 D_{ij} 表示结点 i 和 j 之间的数据传输量,CC_{ij} 表示结点 i 和 j 之间的传输通信代价。

传输数据量的估计依赖于结点 i 参与查询处理的关系的大小和本地/连接操作的选择。例如,如果在结点 i 仅访问关系 R_r 并将它传给下一个结点 j 参与连接处理,这时 D_{ij} 的计算公式为:

$$D_{ij}=SR_r*\prod_{p\in L_p}SEL_p$$

其中,SR_r 为关系 R_r 的大小,L_p 为本地处理谓词集,SEL_p 为谓词 p 的选择性。

如果结点 i 连接了两个关系 R_r 和 R_d 并将连接结果传给了结点 j,这时传输的数据量

$$D_{ij}=SR_r*SR_d*SEL_p$$

4.2 并行处理算法

对于按前文所述方法已确定好执行计划的查询,本文采用如下方法进行并行处理:如果执行计划路径上某一条边的后续边的归一化数据传输代价 NDTC[6](Normalized Data Transmission Cost)(数据传输代价/边的通信代价)高于该边的归一化数据传输代价,则执行计划路径就在该边上断开;然后后面的子路径按照同样的方式进行处理一直到整个执行计划路径结束为止。由于在初始执行计划路径上可能存在多个满足上述标准化据传输代价的边集,所以该方法最终会出现多个并行子路径。并行处理的函数如下(假定执行计划中各个结点已经处理为有序的序列,表示为 $N_1,N_2,N_3,\cdots$):

```
/* 并行处理函数,m 表示开始结点下标,n 表示结束结点下标(m<n),SP 存储子路径集 */
void parallel(int m,int n)
{
  int i,j;
  i=m;
  j=n;
  if(j-i≥2)do
  {
   for(;i≤j-2;i++)
   {
    计算边 N_iN_{i+1} 的 NDTC_{i,i+1} 和 N_{i+1}N_{i+2} 的 NDTC_{i+1,i+2};
     if(NDTC_{i+1,i+2}<NDTC_{i+1,i+2})do
      {将{N_mN_{m+1},…,N_{i+1}}加入 SB;
       m=i+2;
       parallel(N_m,N_n);
       break;
      }
   }
   else
    将{N_m,N_n}加入 SB;
}
```

5 实验

为了测试 GSQO 的有效性,在模拟网格环境中分别采用 GSQO 和 RO(Randomized Optimizer)对同一查询集(3 个查询)进行了查询处理。模拟网格环境如下:(1)利用 Globus Toolkit 构建了一个网格系统;(2)网格中的结点由 12 台主机组成,各

台主机通过局域网连接,它们的配置见表1;(3)创建了4个数据库,每个数据库所包含的关系和副本存储的位置见表2;(4)针对4个数据库构建了一个全局本体,存储于网格服务器中。RO不同于GSQO,在处理过程中没有使用本体查询优化方法和并行处理策略,资源选择的策略是随机地从候选主机中为每一个基本关系选择主机。

表1 网格结点配置

主机名	RAM/MB	MIPS
Host1	1 024	654
Host2	512	612
Host3	1 024	836
Host4	256	400
Host5	128	400
Host6	512	654
Host7	512	612
Host8	1 024	836
Host9	1 024	736
Host10	2 048	1 674
Host11	256	416
Host12	128	328

表2 数据库包含的关系和副本存储的位置

数据库名	包含的关系	副本位置
DB1	*R*1,*R*3	Host1,Host2,Host3
DB2	*R*5	Host4,Host5,Host6,Host7
DB3	*R*4	Host8,Host9,Host10
DB4	*R*2	Host11,Host12

为了检测查询中各个阶段的时间花费,将整个查询处理时间(QPT)分3个部分:QSET(查询语义扩展时间)、QOT(查询优化时间)和QET(查询处理时间)。在实验中提供了3个查询*Q*1、*Q*2和*Q*3,分别使用GSQO和RO对每一个查询进行了三次处理,各个阶段所花费的平均时间及QPT见表3。

表3 *Q*1、*Q*2和*Q*3分别使用RO和GSQO在各个阶段的时间开销

查询名	优化方法	平均QSET/ms	平均QOT/ms	平均QET/s
*Q*1	RO	4 650	19	38
*Q*1	GSQO	2 200	36	35
*Q*2	RO	3 360	15	36
*Q*2	GSQO	1 784	26	29
*Q*3	RO	1 670	8	24
*Q*3	GSQO	880	14	22

从表3可知,各个查询中RO方法花费的平均QOT要小于GSQO方法,但RO方法所花费的平均QSET和QET却大于GSQO,最后RO所花费的整个QPT时间明显要高于GSQO,这主要归功于GSQO在查询中对用户查询扩展、主机选择和并行处理进行了优化处理,尽管GSQO在优化过程中需要花费较多的时间,但为整个查询响应时间的减少打下了基础。实验结果表明,在网格环境中采用本文提出的GSQO方法能减少查询响应时间,获得较好的查询效率。

6 总结

在基于网格的数据库集成环境中,将本体技术运用到信息查询中,构造基于本体的语义查询较好地解决了传统信息查询中由于缺乏语义支持所造成的问题。为了提高基于本体的网格数据查询效率,提出了一个查询优化器—GSQO,该优化器主要由用户查询语义扩展、资源选择和并行处理模块组成,用户查询语义扩展模块主要采用基于BGP(基本图模式)的SPARQL查询优化策略对本体进行查询,实现对用户查询的快速语义扩展;资源选择模块的输入为AQT,主要是采用一个融入了几个效率相关因数的排位函数,对于一个特定的操作按照有效资源所需花费的代价进行排位最后选择花费代价最小的资源执行该操作;并行处理模块通过计算归一化数据传输代价](Normalized Data Transmission Cost)(数据传输代价/边的通信代价)来实现查询计划的并行处理。在模拟网格实验环境中,对GSQO和RO方法在查询各个阶段所花费的时间进行了比较,实验结果表明GSQO提供了较好的查询效率。

参考文献:

[1] 邓志鸿,唐世渭,张铭,等.Ontology综述研究[J].北京大学学报:自然科学版,2002,38(5):730-738.

[2] Berman F,Wolski R.The AppLeS project:A status report[C]//Proc of 8th NEC Research Symposium,Berlin,1997.

[3] Raman R,Livny M,Solomon M.Matchmaking:Distributed resource management for high throughput computing[C]//Proc of 7th IEEE International Symposium on High Performance Distributed Computing,Chicago,1998.

[4] Andrade H,Kurc T,Sussman A,et al.Active Proxy-G:Optimizing the query execution process in the grid[C]//The IEEE/ACM SC2002 Conference,2002:14.

[5] Alpdemir N M,Mukherjee A,Gounaris A,et al.OGSA-DQP:A grid service for distributed querying on the grid[C]//Proc of 9th International Conference on Extending Database Technology,EDBT,2004:858-861.

[6] Srikumar Krishnamoorthy,Avdhoot Kishore Saple.An integrated query optimization system for data grids[C]//Proc of the 1st Bangalore Annual Compute Conference,New York,2008:324-332.

[7] Liua S,Karimib H A.Grid query optimizer to improve query processing in grids[J].Future Generation Computer Systems,2008,24(5):342-353.

[8] Prud'hommeaux E,Seaborne A.SPARQL query language for RDF[EB/OL].[2008].http://www.w3.org/TR/rdf-sparql-query/.

[9] Stocker M,Seaborne A,Bernstein A.SPARQL basic graph pattern optimization using selectivity estimation [C]//Proceeding of the 17th International Conference on World Wide Web,Beijing,2008:595-604.

[10] Ioannidis Y E.The history of histograms (abridged)[C]//Proc of the Intl Conf on Very Large Databases(VLDB),Berlin,2003:19-30.

[11] Mackert F L,Lohman M G.R* optimizer validation and performanceevaluation for distributed queries[C]//The 12th International Conference on Very Large Data Bases,Kyoto,1986.