SMJ:基于大纲的数据流多连接操作

尹田田[1], 张俊虎[2]
(1. 中国科学院 自动化研究所 中法联合实验室, 北京 100190; 2. 青岛理工大学 计算机学院, 山东 青岛 266033)

摘 要: 研究了 Ad hoc 网络的特点及应用情况,分析了在 Ad hoc 网络中传统数据流多连接查询处理策略所面临的问题,并在此基础上提出了一种基于大纲的多数据流连接优化算法(SMJ)。实验结果显示 SMJ 算法可以极大降低 Ad hoc 网络数据流连接查询处理的通信代价。

关键词: 自组织网络; 大纲; 数据流; 多连接; 多数据流连接算法
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2009)05-1847-05

SMJ:synopsis-based data stream multi-join operation

YIN Tian-tian[1], ZHANG Jun-hu[2]
(1. *LIAMA, Institute of Automation, Chinese Academy of Sciences, Beijing* 100190, *China*; 2. *School of Computer Science & Engineering, Qingdao Technological University, Qingdao Shandong* 266033, *China*)

Abstract: This paper studied the characteristics and the applications of Ad hoc networks, and analyzed the problems of the traditional methods of multi-stream join processing in Ad hoc networks. Based on this, proposed a synopsis-based multi-stream join optimization algorithm——SMJ. The experiments show that this method can greatly reduce the communication cost.
Key words: Ad hoc network; synopsis; data stream; multi-join; SMJ algorithm

0 引言

Ad hoc 网络是一种逻辑意义上的组网方式,是在不依赖基础网络设施的前提下由一定范围内的移动终端动态地建立可以互连的网络。它在军事、基于无线传感器网络(WSNs)的应用、事件监测系统、无线 P2P 应用等领域使用广泛。

Ad hoc 网络具有以下主要特点:

a)无中心。Ad hoc 网络是一个对等式网络,节点可以随时加入或离开网络。

b)自组织。网络的布设或展开无须依赖于任何预设的网络设施。

c)多跳路由。当节点要与其覆盖范围之外的节点进行通信时,需要中间节点的多跳转发。

d)动态拓扑。Ad hoc 网络中节点位置可以改变。节点可以随时加入或离开网络,从而使网络拓扑结构动态变化。

数据流是一种持续产生的数据元组的序列。在 Ad hoc 网络中,许多应用需要处理多数据流连接查询。例如由传感器组成的 Ad hoc 网络可用于监测某个区域动物活动情况。网络中的每个节点都能记录经过本节点的动物编号,并且每个节点都能知道自己的地理位置。网络上任意节点可查询:在三个不同位置的节点 $site_1$、$site_2$、$site_3$ 都出现过动物的编号,本文用 R_1、R_2、R_3 分别表示 $site_1$、$site_2$、$site_3$ 三个节点上的数据流。R_1、R_2、R_3 的模式为{ID,time}。其中:ID 为某一动物的惟一标志;time 为该动物出现的时间,即元组的时间标签。数据流的查询可以表示成一个多连接 R_1 join R_2 join R_3。

为优化该查询,以查询处理过程中所产生的通信代价作为优化目标。通信代价是一个查询处理过程中单位时间内网络任意两个节点之间所传输数据量的总和。

1 相关工作

最近出现了很多数据流管理系统(DSMS),如面向监控应用数据流处理的 Aurora[1] 系统;面向电信记录数据流处理的 TelegraphCQ[2] 系统;面向传感器网络数据流处理的 Fjord[3] 系统等。还有很多文章讨论了多数据流连接查询的自适应处理技术。Eddy[4] 是一种数据流查询机制,可以在集中式的环境下持续不断地为多连接查询操作符重新排序,以提高结果输出速度。A-Greedy[5] 算法是一种动态改变多连接执行顺序以减少中心节点计算代价的算法。Joseph 等人[6] 提出了自适应地调整查询树以获得数据流多连接最大化输出速度的技术。Tian 等人[7] 解释了动态查询计划迁移的具体算法。

以上系统是将所有数据流传送到一个节点处理,没有考虑分布在不同数据源节点上的多数据流连接查询通信代价问题。在 Ad hoc 网络中,通信所产生的能耗远远高于本地计算产生的能耗,降低通信代价尤为重要。D-eddy[8] 和 SwAP[9],实现了分布式的多数据流连接查询。这两者使用 Lottery routing 策略,启发式地决定分布式数据流上多连接执行顺序。但是前者优化查询处理的响应时间,后者优化后的通信代价依然很高。

基于以上问题,本文提出了一种基于大纲的多数据流连接优化算法 SMJ。通过大纲信息进行多数据流连接查询,能避免传输不能产生最终结果的数据元组,并在每个连接时段优化多数据流连接执行顺序。实验显示 SMJ 算法与 Lottery routing 算法相比能更大程度地降低多数据流连接查询的通信代价。

收稿日期: 2008-08-02; **修回日期**: 2008-09-10

作者简介:尹田田(1983-),男,北京人,硕士研究生,主要研究方向为数据库(9997766@163.com);张俊虎(1974-),男,山东青岛人,副教授,博士后,主要研究方向为分布式数据库、数据网络.

2 多数据流连接的执行顺序

$R_1,\cdots,R_n$ 是数据源节点 $site_1,\cdots,site_n$ 上的数据流。若要执行多数据流连接 R_1 join R_2 join…join R_n,则每个数据源节点 $site_i$ 上$[t_{j-1},t_j]$时段新产生的数据块 ΔR_i^{new} 要与其他节点 t_{j-1} 时刻之前产生的数据块 $R_1^{old},\cdots,R_{i-1}^{old},R_{i+1}^{old},\cdots,R_n^{old}$ 依次进行连接,即 ΔR_i^{new} 按照一定次序先传递到节点 $site_1,\cdots,site_{i-1},site_{i+1},\cdots,site_n$ 中的某一个节点上连接,然后将其连接结果传递到另一个节点上作连接,依此类推,直到所有的连接执行完毕,将最终结果传送到查询节点。

按照这种多数据流的连接方式,不同的执行顺序会产生相同的最终结果,但会产生不同的通信代价。下面用上文动物监测的实例来说明这个问题。

表 13 显示了数据流 R_1、R_2、R_3 在 t_1 时刻前的数据。ΔR_2(表 4)是 $site_2$ 节点在$[t_1,t_2]$时段产生的新数据块。$\Delta R'_2$(表 5)是 $site_2$ 节点在$[t_2,t_3]$时段产生的新数据块。表 6 显示了 $site_1$,$site_2$,$site_3$ 之间的单位数据通信代价。对于 $site_2$ 上每块新数据,都有两种不同的执行顺序进行连接操作。例如,ΔR_2 的连接可有以下两种方案:

a)首先传递到 $site_1$ 以完成 ΔR_2 join R_1,然后将中间结果 ΔR_2 join R_1 传递到 $site_3$ 以完成(ΔR_2 join R_1) join R_3。

b)首先传递到 $site_3$ 以完成 ΔR_2 join R_3,然后将中间结果 ΔR_2 join R_3 传递到 $site_1$ 以完成(ΔR_2 join R_3) join R_1。

假定 w_{ij}代表节点 $site_i$, $site_j$ 之间的单位数据通信代价。$|R|$代表 R 的大小。那么上述两种情况所产生的通信代价分别为

方案 1 代价:$|\Delta R_2|\times w_{21}+|\Delta R_2 \text{ join } R_1|\times w_{13}$;

方案 2 代价:$|\Delta R_2|\times w_{23}+|\Delta R_2 \text{ join } R_3|\times w_{31}$。

根据表 6 所示的通信代价矩阵,对于 ΔR_2,两种执行顺序所产生的通信代价分别为:a)$3\times18+3\times25=129$;b)$3\times20+2\times25=110$。所以,对于 ΔR_2 最优的执行顺序是第二种。

对于 $\Delta R'_2$,两种不同的执行顺序的通信代价分别为:a)$3\times18+2\times25=104$;b)$3\times20+2\times25=110$。那么 $\Delta R'_2$ 的最优执行顺序是第一种。

表 1 $site_1$ 在 t_1 时刻的数据

ID	time
3	00:10
4	00:13
2	00:23
3	00:34

表 2 $site_2$ 在 t_1 时刻的数据

ID	time
10	00:10
6	00:17
1	00:23

表 3 $site_3$ 在 t_1 时刻的数据

ID	time
8	00:10
4	00:13
5	00:15
3	00:16

从上面的例子可以看出,针对不同时段的不同数据,应采用不同的执行顺序,以降低通信代价。最优的执行顺序是随时间变化的。要降低通信代价,必须在不同时段按照最优的执行顺序执行多连接操作。

通过上面的分析,可知,影响数据流多连接通信代价大小因素有以下三个:

a)连接属性值的重叠度。两个数据表之间的重叠度用它们之间在连接属性上相同取值的个数表示。重叠度越大,两数据表在进行连接操作所产生的结果越多。

b)数据流数据产生的速度。数据源节点产生数据的速度越快,单位时间内产生的数据量越多,单位时间内产生的结果也越多。

c)数据节点之间拓扑结构的变化。两个节点之间单位数据传输代价越大,传输同样数据的通信代价越大。

为了以最小的通信代价实现多数据流连接查询处理,一个好的处理策略应该能够针对当前时段多数据流上连接属性值的重叠度、数据源节点之间拓扑结构的变化。各数据流上数据产生的速度,产生相应的最优执行顺序。为此,将多数据流的持续连接转换为多个连接时段内的多个数据块间的连接,并引入大纲信息来处理多数据流连接优化。通过大纲信息可避免传输新生成数据块中不能产生最终结果的元组,并在每个时段,为能够产生最终结果的元组确定最优执行顺序。

表 4 $site_2$ 在$[t_1,t_2]$时段产生的新数据

ID	time
4	00:46
8	00:56
3	00:70

表 5 $site_2$ 在$[t_2,t_3]$时段产生的新数据

ID	time
5	00:75
3	00:90

表 6 单位数据通信代价矩阵

src\des	1	2	3
1	0	18	25
2	18	0	20
3	25	20	0

3 基于大纲的多数据流连接算法 SMJ

本章将详细描述基于大纲的多数据流连接优化算法 SMJ。假设要处理的查询 $Q=R_1$ join R_2 join… join R_n。其中:$R_1\cdots R_n$ 是来自节点 $site_1,\cdots,site_n$ 上的数据流。

3.1 基于大纲中心的多连接执行顺序的优化

基本定义如下:

a)大纲向量。它是一个 $m\times1$ 维的数据向量。设 S 表示数据流 R 在一定时间间隔之内的大纲向量。S 中的每一项是在属性 A 上具有一个特定的值的元组的数目,m 对应 A 上所有可能取值的个数。设属性 A 总共有四个可能值 v_1、v_2、v_3、v_4(升序排列),这四个不同的属性值上所产生元组的数目分别是 3、7、9、1,R 在属性 A 的大纲向量就是一个 4×1 维的数据向量$S=[3,7,9,1]$。表 7 是 R_1、R_2、R_3 在连接属性 ID 上的大纲向量,这里假设连接属性的可能取值为 110,S_i 表示 R_i 的大纲向量。在以后的叙述中用 $S_i(a)$ 来表示 R_i 中属性 A 值为 a 元组数目。

b)增量大纲向量。每个数据流上产生的新数据块的大纲向量叫做增量大纲向量。如表 4 所示的新数据块 ΔR_2,其增量大纲向量 $\Delta S_2=[0,0,1,1,0,0,0,1,0,0]$,这是因为 ΔR_2 在连接属性 ID = 3,4,8 上各产生一个元组,因此其增量大纲向量 ΔR_2 在其 3、4、8 位置上的分量的取值为 1,其他为 0。

c)通信代价矩阵。数据源节点之间单位数据传送代价构成的矩阵,记做通信代价矩阵 W,其元素 w_{ij}是节点 $site_i$ 与 $site_j$ 之间传送单位数据的代价,$1\leqslant i\neq j\leqslant n$,$n$ 为数据源节点个数。

d)$\odot$为二元矩阵元素乘的算子。若向量 $a=[a_1,a_2,\cdots,a_n]$,向量 $b=[b_1,b_2,\cdots,b_n]$,则 $a\odot b=[a_1*b_1,a_2*b_2,\cdots,a_n*b_n]$。

e)$\oplus$为一元矩阵元素加的算子。若 $v=[2,7,3,4]$,则 $v^{\oplus}=2+7+3+4=16$。

表 7 R_1、R_2、R_3 在 t_1 时刻的大纲向量

ID	S_1	S_2	S_3	ID	S_1	S_2	S_3
1	0	1	0	6	0	1	0
2	1	0	0	7	0	0	0
3	2	0	1	8	0	0	1
4	1	0	1	9	0	0	0
5	0	0	1	10	0	1	0

使用大纲信息和通信代价矩阵,可以按照如下方式确定某

执行顺序的通信代价。如第 2 章提到对新数据块 ΔR_i^{new} 的多连接:ΔR_i^{new} join R_1^{old} join R_2^{old} join…join R_{i-1}^{old} join R_{i+1}^{old} join …join R_n^{old} 设一个可能的执行顺序为 $R_{k\_1}^{old},\cdots,R_{k\_n-1}^{old}$。其中 $k_1,\cdots,k_{n-1}$ 是集合 $\{1,\cdots,i-1,i+1,\cdots,n\}$ 的一个排列。用 $\Delta S'_i$ 表示 ΔR_i^{new} 经过大纲中心处理过的增量大纲向量(后文称做滤波大纲向量)。要完成这个多连接操作,首先将 ΔR_i^{new} 的 $\Delta S'^{\oplus}_i$ 个元组传送到 $site_{k\_1}$ 完成 ΔR_i^{new} join $R_{k\_1}^{old}$,通信代价为 $\Delta S'^{\oplus}_i * w_{i,k\_1}$。中间结果为 $\Delta S'^{\oplus}_i$ 个元组,被传送到 $site_{k\_2}$ 完成 ΔR_i^{new} join $R_{k\_1}^{old}$ join $R_{k\_2}^{old}$,通信代价为 $(\Delta S'_i \odot S_{k\_1})^{\oplus} w_{k\_1,k\_2}$。中间结果为 $(\Delta S'_i \odot S_{k\_1} \odot S_{k\_2})^{\oplus}$ 个元组。如此反复,直到 $(\Delta S'_i \odot S_{k\_1} \odot S_{k\_2} \odot \cdots \odot S_{k\_n-2})^{\oplus}$ 个元组被传送到 $site_{k_{n-1}}$ 来完成最后的连接,通信代价为 $(\Delta S'_i \odot S_{k\_1} \odot S_{k\_2} \odot \cdots \odot S_{k\_n-2})^{\oplus} * w_{k\_n-2,k\_n-1}$。去除将最终结果传送给查询节点的通信代价,多数据流连接 ΔR_i^{new} join $R_{k\_1}^{old}$ join … join $R_{k\_n}^{old}$ 产生的总通信代价为 $\Delta S'^{\oplus}_i * w_{i,k\_1} + (\Delta S'_i \odot S_{k\_1})^{\oplus} w_{k\_1,k\_2} + \cdots + (\Delta S'_i \odot S_{k\_1} \odot \cdots \odot S_{k\_n-2})^{\oplus} * w_{k\_n-2,k\_n-1}$。

如上所述,使用大纲信息和数据源节点之间通信代价矩阵可确定某个执行顺序的通信代价,可以据此选出不同的执行顺序中通信代价最小的执行顺序。仍然是用上文提到的用动物习性监测的例子来说明,如何使用大纲信息去除不能产生最终结果的元组和确定最优的执行顺序。

从表 7 中可以知道 ΔR_2(表 4)中 ID = 3 或 4 的元组可以产生最终的结果(表 7 中所有 $S_i(3), S_i(4)$ 都不为 0,$i = 1,3$)。而 ID = 8 的元组不能产生最终的结果($S_1(8) = 0$),滤掉 ΔR_2 中 ID = 8 的元组。表 7 结合表 6 的信息,可以计算 ΔR_2 按照不同的执行顺序来进行连接操作产生的通信代价。

如果 ΔR_2 按照第 2 章中方案 2 来执行,先把 ΔR_2 中 ID = 3,4 的元组发到 $site_1$ 来完成 $J_1 = \Delta R_2$ join R_1。ΔR_2 中 ID = 3,4 的元组有 $\Delta S_2(3) + \Delta S_2(4) = 2$ 个,这个过程产生的通信代价是 $2 \times 18 = 36$,所产生的结果 J_1 有 $\Delta S_2(3) \times S_1(3) + \Delta S_2(4) \times S_1(4) = 1 \times 2 + 1 \times 1 = 3$ 个元组。将 J_1 从节点 $site_1$ 传送到 $site_3$ 完成 J_1 join R_3。这个过程产生的通信代价是 $3 \times 25 = 75$。所以总的代价是 $36 + 75 = 111$。

如果 ΔR_2 按照方案 2 来执行,先把 ΔR_2 中 ID = 3,4 的元组发到 $site_3$ 来完成 $J_1 = \Delta R_2$ join R_3。ΔR_2 中 ID = 3,4 的元组有 $\Delta S_2(3) + \Delta S_2(4) = 2$ 个,这个过程产生的通信代价是 $2 \times 20 = 40$,所产生的结果 $J_1 = \Delta R_2$ join R_3 有 $\Delta S_2(3) \times S_3(3) + \Delta S_2(4) \times S_3(4) = 1 \times 1 + 1 \times 1 = 2$ 个元组。将 J_1 从节点 $site_3$ 传送到 $site_1$ 完成 J_1 join R_1。这个过程产生的通信代价是 $2 \times 25 = 50$。所以总的代价是 $40 + 50 = 90$。

基于大纲的方法最优通信代价是 90,而不使用大纲的通信代价是 110(ΔR_2 按照方案 2 的通信代价),比原来减少为 $110 - 90 = 20$。

综上所述,通过大纲的方式处理多连接优化具有以下特点:

a)去除不能产生最终连接结果的元组,减少数据传输量。

b)利用大纲信息,能够描述每个数据流在连接属性值上的统计信息,为每个时间段产生的数据确定最优的连接执行顺序。

c)对每个时间段产生的数据,针对连接属性值重叠度,网络拓扑和数据产生速度的变化自适应地确定最优执行顺序。

3.2　SMJ 算法基本思想

本节介绍基于大纲的多数据流连接优化算法 SMJ。首先引入大纲中心的概念:

大纲中心——到网络中所有数据源节点的距离和最小的节点。它维护并不断更新多数据流连接查询中所有数据流的大纲向量,动态地获取数据源节点之间的拓扑信息。通过这些信息,大纲中心动态地为每个数据流上的每块新数据确定一个最优的执行顺序。新数据按照这个数据依次路由到不同的节点上进行连接操作,完成最终结果。它维护两种信息,即所有数据流的大纲向量 $S_1,\cdots,S_n$ 数据源节点之间的单位数据通信代价矩阵 W。

基于大纲的多数据流连接查询处理算法 SMJ 的工作机制如下:

a)某一个数据源节点 $site_i$ 上的数据流产生新的数据块时,该数据块被缓存,相应的增量大纲向量被发送到大纲中心,用于大纲中心作更新。如表 4 所示的新数据块 ΔR_2,其增量大纲向量为 $\Delta S_2 = [0,0,1,1,0,0,0,1,0,0]$。

b)当大纲中心收到来自数据源节点 $site_i$ 的增量大纲向量时,执行:(a)更新相应数据流的大纲向量;(b)根据更新的增量大纲向量产生滤波大纲向量;(c)根据更新滤波向量产生最优执行顺序;(d)向节点 $site_i$ 返回最优执行顺序和滤波向量。例如,对表 4 所示的新数据块 ΔR_2,大纲中心收到 ΔS_2 后进行更新操作:$S_2 \leftarrow S_2 + \Delta S_2$。将 ΔS_2 与其他大纲向量作元素乘操作,得到滤波向量 $\Delta S'_2 \leftarrow \Delta S_2 \odot S_1 \odot S_3 = [0,0,1,1,0,0,0,0,0,0]$。其中不为 0 的元素即为能产生最终结果的元组。为 $\Delta S'_2$ 确定最优的执行顺序 $\{3,1\}$(算法将在 3.3 节详细解释)。将此执行顺序附在 $\Delta S'_2$ 后面发送回 $site_2$。

c)当节点 $site_i$ 收到大纲中心返回的滤波大纲向量和执行顺序后,节点首先根据滤波向量从新数据块中去除不能产生最终结果的元组,将剩余元组按照指定的顺序依次发送到不同的节点上完成连接,并将最终结果发送到查询节点。例如,$site_2$ 收到 $\Delta S'_2$ 和执行顺序 $\{3,1\}$ 后,先半连接操作 $\Delta R_2 \leftarrow \Delta R_2$ semi-join $\Delta S'_2$,然后 ΔR_2 依次被发送到 $site_3, site_1$ 作连接以完成最终结果。

3.3　SMJ 算法详细描述

根据以上的分析,得出基于大纲的最优执行顺序选择算法 SMJ-E。这种算法穷举每一种可能的执行顺序,从中选出代价最小的执行顺序。SMJ-E 算法如下:

```
输入:通信代价矩阵 W,大纲向量集 S1,S2,…,Sn
输出: 执行顺序 orderD
begin
  for ΔR_i^new, ∀ i ∈ 1…n
    Generate synopsis vector ΔS_i for ΔR_i^new;
    minimalcost←∞;
    cost←0;
    orderD←{0,0,…,0};
    {k1,k2,…,k_n-1}←{0,0,…,0};
    for(j = 1 to n!)
      {k1,k2,…,k_n-1}←newpossibleorder();
      //产生{k1,k2,…,k_n-1}全排列中的下一个排列
      cost←ΔS'_i^⊕ * w_i,k_1 + (ΔS'_i ⊙S_k_1)^⊕ * w_k_1,k_2 + … +
(ΔS'_i ⊙S_k_1 ⊙…⊙S_k_n-2)^⊕ * w_k_n-2,k_n-1;
      if(minimalcost < cost)
        cost? minimalcost;
        orderD←{k1,k2,…,k_n-1};
      endif
```

```
    endfor
  endfor
end
```

由于算法 SMJ-E 穷举出每一种可能的执行顺序,它的计算复杂度为 $O(n!)$。本文提出了一种基于动态规划的启发式算法 SMJ-H,算法复杂度只有 $O(n^2)$。SMJ-H 算法如下:

```
输入:通信代价矩阵 W,大纲向量集 S1,S2,…,Sn
输出: 执行顺序 orderD
begin
  for ΔRi^new, ∀ i ∈ 1…n
    Generate synopsis vector ΔSi for ΔRi^new;
    orderD←∅;
    orderX←{1,2,…,i-1,i+1,…,n};
    b←ΔSi;
    for(i=2 to n-1)
      d←b⊙S_orderX(1);
      m←orderX(1);
      for(j=2 to n-i)
        if(d > (b⊙S_orderX(j))⊕
          d←(b⊙S_{orderX(j)})⊕;
          m←orderX(j);
        end if
      endfor
      orderX←orderX - orderX(m);
      orderD←orderD + orderX(m);
      b←b⊙S_orderX(m);
    endfor
  endfor
end
```

4 基于滑动窗口的多数据流连接优化

数据流具有无界的特性,多数据流连接应用采用滑动窗口技术将无界的数据流连接转换成连续的有界数据集的连接。其中有界数据集由当前时间推后 T_w 时间(时间窗)内的数据组成[10]。如算法 SMJ 仅需少许修改,即可适应基于滑动窗口的多数据流连接查询。

当某数据流产生了新数据块时,会有一些旧元组 r 的时间戳 t_r 超出窗口 T_w,$|t_c - t_r| > T_w$。其中,t_c 为当前时间戳。这种位于时间窗外的数据将被删除。某数据流中有数据被删除时,为已删除数据建立减量大纲向量,并将其发送到大纲中心作相应更新。减量大纲向量是已删除数据的大纲向量,当大纲中心收到减量大纲向量时,对相应数据流的大纲向量作减操作。

例如,对第 2 章提到的例子,设数据流 R_2 的时间窗 T_w 为 2 个时间单位,那么当$[t_2, t_3]$时段的数据块 $\Delta R'_2$ 产生时(设 $|t_2 - t_1|$,$|t_3 - t_2|$均为 1 个时间单位),t_2 之前的数据,时间戳超出时间窗($|t_3 - t_1| = T_w$),应该删除,即表 2 的数据要被删除。故 $\Delta R'_2$ 产生时,节点 $site_2$ 向大纲中心发送两种大纲向量:

a)减量大纲向量。$\Delta^- S'_2 = [1,0,0,0,0,1,0,0,0,1]$,表示数据流 R_2 中当 $\Delta R'_2$ 产生时要删除数据(表 2 的数据)的大纲向量。

b)增量大纲向量。$\Delta^+ S'_2 = [0,0,1,0,1,0,0,0,0,0]$,表示数据流 R_2 新产生数据(表 5 的数据)的大纲向量。

$site_2$ 将 $\Delta^- S'_2$ 和 $\Delta^+ S'_2$ 一起发给大纲中心,大纲中心进行更新操作 $S_2 \leftarrow S_2 - \Delta^- S'_2 + \Delta^+ S'_2$,然后,大纲中心按照第 3 章提到的算法进行处理。

5 实验及结果分析

5.1 实验设置

假设存在一个配置在一个区域内的监测网络,随机选择其中的几个节点作为数据源节点,满足:

a)每个数据源节点对应一个数据流。

b)任意两数据源节点间的单位数据通信代价是一个 110 的随机数。

c)每个数据流单位时间产生的元组个数,符合参数 $\lambda = 5$ 的泊松分布。

d)所有连接都在同一个属性上,连接属性值符合 120 的均匀分布。

e)所有数据流的运行时间都是 30 个时间单位,时间窗为 3 个时间单位。

f)设一个大纲向量元组为一个普通数据元组的 1/4。

5.2 实验算法

基于大纲的多数据流连接优化算法 SMJ 在优化多连接执行顺序时可采用两种方式:a)启发式,称之为 SMJ-H 算法;b)穷举式,称之为 SMJ-E 算法。将其与 Lottery routing[4] 算法比较,以测试 SMJ 算法性能。

Lottery routing 算法在 D-eddy[8] 和 SwAP[9] 得以在分布式环境中实现。这种算法为某个数据源节点产生的新数据块,引入历史向量 h。其中 h 由 n 个 bit 位表示,n 是数据源节点个数,每个 bit 位表示这块数据是否已被某个数据源节点处理过,是则为 0;否则为 1。每个数据源节点 $site_k$ 维护一个到其他数据源节点的选择因子向量 $sf = \{sf_1^k, \cdots, sf_{k-1}^k, sf_{k+1}^k, \cdots, sf_n^k\}$。其中:$sf_i^k$ 是节点 $site_k$ 向节点 $site_i$ 发送数据元组个数和节点 $site_i$ 产生的相应结果元组个数的比率。节点 $site_k$ 根据每次向 $site_i$ 发送的数据量以及 $site_i$ 反馈给 $site_k$ 产生的结果数动态修改 sf_i^k。对于 $site_k$ 上一块具有历史向量 h 的数据,它选择向量 $sf \odot h$ 中最大元素对应的节点作为下一个要传送到的节点。

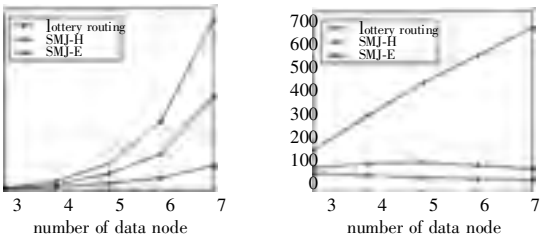
5.3 结果分析

实验中,本文研究了无滑动窗口和有滑动窗口条件下进行多数据流连接时,不同算法的单位时间通信代价随时间积累和数据源节点数目增加的变化情况。

图 1(a)和(b)分别显示了无滑动窗口和有滑动窗口条件下,算法 Lottery routing、SMJ-H 和 SMJ-E 在进行多数据流连接时所产生的单位时间通信代价与数据源节点数目之间的关系:无论是否基于滑动窗口,多数据流连接所需的中间连接结果的传送次数会随数据源节点数目的增加而增多,导致三个算法进行多数据流连接时的单位时间通信代价一致增大。其中 Lottery routing 计算多数据流连接所产生的单位时间通信代价最大,而 SMJ-H 所产生的单位时间通信代价又略高于 SMJ-E。另外,由图 1(b)可知,在基于滑动窗口的多数据流连接优化方面,随着数据源节点数目的增多,SMJ-H/E 在减少通信代价方面的优势尤为明显。

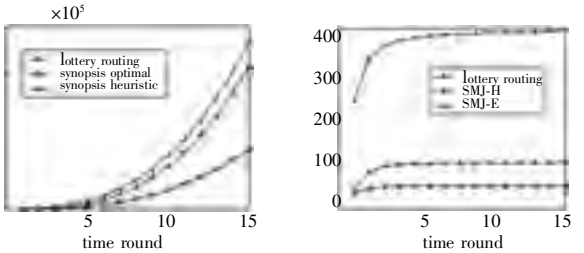
图 2(a)和(b)分别显示了非滑动窗口连接和滑动窗口连接的情况下,单位时间通信代价随时间的变化。实验结果显示,非滑动窗口多连接查询,单位时间通信代价随着时间增加而增加。这是因为对非滑动窗口连接,每个数据源节点上的数

据随时间逐步增多,因此参与计算的数据量逐步增大,从而使单位时间通信代价增加。对于基于滑动窗口的连接,开始时单位时间通信代价随时间增加而增加,而后单位时间通信代价趋向一个常值。这是因为开始时,每个数据源节点上的数据量未满一个窗口,此时随时间增长,节点上积累的数据增多;当数据源节点中的数据开始超过一个窗口时,旧数据会被删除,使等待处理的数据量不会变化太大,从而使单位时间通信代价趋于常值。



(a) 无滑动窗口多连接,单位时间通信代价随数据源节点数目的变化　(b) 滑动窗口多连接,T_w=3,单位时间通信代价随数据源节点数目的变化

图 1　有无滑动窗口通信代价变化



(a) 无滑动窗口多连接,单位时间通信代价随时间的变化(数据源节点数目 5)　(b)滑动窗口多连接,T_w=3 时间单位,单位时间通信代价随时间的变化(数据源节点数目 5)

总之,基于大纲的多连接执行顺序优化算法 SMJ 在降低通信代价方面表现出色:算法 SMJ-H 在非滑动窗口多连接时,比 Lottery routing 算法降低 20%40% 的通信代价;在滑动窗口连接的情况下可以降低 50%70% 的通信代价。算法 SMJ-E 在非滑动窗口多连接时,比 Lottery routing 算法降低 40%60% 的通信代价;在滑动窗口多连接的情况下降低 60%85% 的通信代价。

6　结束语

本文研究了在 Ad hoc 网络环境下数据流多连接查询处理。首先分析了数据流多连接不同的执行顺序对通信代价的影响;然后分析了影响数据流多连接通信代价的三个因素,即连接属性值的重叠度、数据流数据产生的速度、数据节点之间拓扑结构的变化。在此基础上,提出了一种基于大纲多数据流连接优化算法 SMJ 来优化数据流多连接查询的执行顺序,并且与 Lottery routing 算法进行比较。实验结果显示,SMJ 能够很好地优化数据流多连接执行顺序,提高 Ad hoc 网络环境中的多连接查询执行效率。

致谢　本文是在中国科学院自动化所中法联合实验室(LIAMA,CASIA)的 NETQUET 项目组中完成的。LIAMA、中法信息、自动化与应用数学联合实验室,由中国科学院和法国计算机研究院等多家顶级研究机构联合创办,是国家级国际联合研究中心之一。NETQUEST 组主要研究无线网络分布式查询处理。感谢 Stéphane Grumbach 和 Christophe Bobineau 两位法国教授的指导。Stéphane Grumbach 是中法实验室法方主任,博士生导师,主要研究方向为数据库理论;Christophe Bobineau 是法国国立高等数学与计算机学院副研究员。

参考文献:

[1] ABADI D J,CARNEY D,ETINTEMEL U C,*et al*. Aurora:a new model and architecture for data stream management[J]. VLDB Journal, 2003,12(2):120-139.

[2] CHANDRASEKARAN S, COOPER O, DESHPANDE A, *et al*. TelegraphCQ:continuous dataflow processing for an uncertain world [C]//Proc of CIDR. 2003.

[3] MADDEN S,FRANKLIN M J. Fjording the stream:an architecture for queries in a production system[C]//Proc of ICDE Workshops. 2002: 555-566.

[4] AVNUR R,HELLERSTEIN J M. Eddies:continuously adaptive query processing[C]//Proc of SIGMOD Conference. 2000:261-272.

[5] BABU S,MOTWANI R,MUNAGALA K,*et al*. Adaptive ordering of pipelined stream filters[C]//Proc of SIGMOD Conference. 2004:407-418.

[6] GOMES J S, CHOI H A. Finding: optimal join tree for multi-join stream queries in a production system[C]//Proc of ICDCS Workshops. 2006.

[7] YANG Yin,KRAMER J,PAPADIAS D,*et al*. HybMig:a hybrid approach to dynamic plan migration for continuous queries[J]. IEEE Trans on Knowledge of Data Engineering, 2007, 19(3):398-411.

[8] FENG Tian,DEWITT D J. Tuple routing strategies for distributed eddies[C]//Proc of VLDB. 2003:334-344.

[9] ZHOU Yong-luan,OOI B C,TAN K L,*et al*. An adaptable distri-buted query processing architecture[J]. Data & Knowledge Engineering, 2005,53(3):283-309.

[10] ZHANG Dong-dong,LI Jian-zhong,KIMELI K,*et al*. Sliding window based multi-join algorithms over distributed data streams[C]//Proc of ICDE. 2006.

(上接第 1846 页)第一类 AQR 算法,就可以通过本文的改进 AQR 模型,并将它应用于第二类 AQR 问题。

参考文献:

[1] 宋乃斌,高随祥,王营昌. 一种基于改进遗传算法的多约束 QoS 路由选择方法[J]. 微型机与应用,2005(8):30-31,61.

[2] WANG Yu,LI Le-min,XU Du. A multi-constrained quality of service routing based on metrics transform[C]//Proc of IEEE International Conference on Networking, Sensing and Control. 2007:525-529.

[3] WANG Yu,LI Le-min,XU Du. Metrics transform based multi-constrained optimal path selection [C]//Proc of IEEE Conference on Communications, Circuits and Systems. 2007:510-514.

[4] HE Rong-xi,LIN Bin,LI Le-min. Dynamic service-level-agreement aware shared-path protection in WDM mesh networks[J]. Journal of Network and Computer Applications,2007, 30(2):429-444.

[5] ARCI D,PELECCHI D,MAIER G,*et al*. Availability models for protection techniques in WDM networks [C]//Proc of the 4th International Workshop on Design of Reliable Communication Networks. 2003:158-166.

[6] MELLO D A,PELEGRINI J U,RIBEIRO R P,*et al*. Dynamic provisioning of shared-backup path protected connections with guaranteed availability requirements[C]//Proc of the 2nd International Conference on Broadband Networks. 2005:1320-1327.