基于层次组织位置数据库的连续位置相关查询处理*)

李国徽
(华中科技大学计算机学院　武汉430074)

Location-Dependent Continuous Query Processing Using Hierarchical Location Databases

LI Guo-Hui
(School of Computer Sci. & Tech. Huazhong Univ. of Sci. & Tech. ,Wuhan 430074)
E-mail:guohuili@public. wh. hb. cn

Abstract With the advances in mobile computing and mobile communication technology, there comes a kind of novel applications in which the locations of moving objects are maintained and processed. In existing literatures, a data model called moving objects sptio-temporal (MOST)[1,2] is proposed and a new location record is generated when the distance between the actual location and the database location of a moving object exceeds a pre-defined distance threshold. In a mobile computing environment, a user can issue location-dependent continuous queries (LDCQs). To cater for the large number of moving objects in the system, this paper first gives a hierarchical distributed location database model to store the locations of moving objects. Based on the distribution of the location databases for different moving objects, this paper then proposes a method to determine the processing site for a location-dependent query. When a LDCQ is processed, a set of tuples ⟨O, begin, end⟩ is provided indicating that object O satisfies the condition presented in the LDCQ from time begin to end. In the existing literatures, when there is a location update generation, the related LDCQ is re-processed and the answering tuples are re-transmitted via the wireless channel. This location-update-based LDCQ processing method has its disadvantages: it has much CPU calculation cost and imposes a high overhead in the wireless bandwidth which is very undesirable in a wirelss environment. Based on the maximal speed of a moving object, this paper presents a deferred LDCQ evaluation strategy.

Keywords Hierarchical location database, Location update generation, Location-dependent queries, Continuous queries, Location management for moving objects

1 引言

现有的数据库系统一般假设数据在未被显式修改前是不变的,例如:如果字段 salary 的值是30,000,那么只有通过事务更新才会改变该字段的值。但对连续变化的对象,如移动对象的位置,应用传统的数据库管理系统来管理会造成两种结果:或者移动对象位置的频繁更新占用大量的系统资源;或者使用移动对象过时的位置信息而导致错误的决策。

文[1,2]中提出了移动对象时空模型(MOST)对移动对象建模。该数据模型中,移动对象不仅有静态属性,还有动态属性。动态属性 A 由三个子属性 A. initialvalue, A. updatetime 和 A. function 组成,即使数据库没有更新,动态属性也会随着时间变化,因此可用来建模随时间连续变化的数据对象。应用 MOST 数据模型,即使数据库未更新,也能估算出 A. updatetime 以后动态属性的值。当然,动态属性的实际值可能和计算值不同,所以系统应对实际值和计算值之间的偏差做相应约束。

目前已有不少关于移动对象数据建模和位置更新策略方面的研究成果[4~6],但在支持连续的位置相关查询(LDCQs)的位置数据库组织和 LDCQs 处理方法的研究工作还很少见到,本文下面部分主要介绍我们在这方面所做的工作。

2 位置数据库组织

2.1 系统模型

目前广泛使用的移动计算系统模型如图1所示,系统由移动主机(mobile hotst,MH)和固定主机(fixed hotst,FH)组成,其中移动主机能够在移动状态下保持网络连接,并通过无线网络和移动支持基站(mobile support station,MSS)进行通信。移动单元是 MSS 所覆盖的一个区域,每个 MH 与一个 MSS 联系并可直接与所在移动单元内的其它 MH 通信。为了与不同移动单元内的 MH 通信,呼叫方 MH 先与本地 MSS 通信,MSS 再通过固定网络将信息传输给被叫方 MH 所在的 MSS。

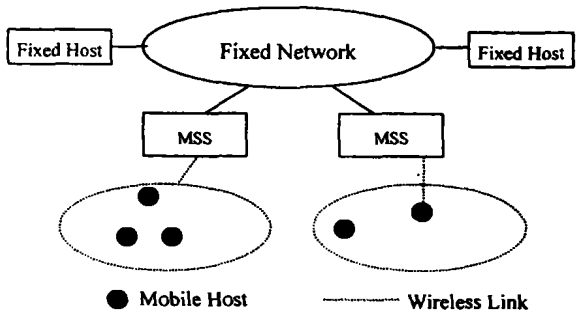

Figure 1 System Architechture of a Mobile Computing System

2.2 数据模型

一般来讲,由于在位置数据库中执行位置的实时更新操作开销太大而不太可能实现,所以系统并不知道移动对象某一时刻的精确位置,因此要有相应的机制估算出移动对象某一时刻的位置。假定系统中所有对象在二维空间运动,基于

*)论文受国家青年自然科学基金(60203017)项目资助。

MOST 数据模型我们给出如下定义：

定义1 一个移动对象的位置记录定义为一个7元组：

〈starttime，x.startposition，y.startposition，direction，speed，maxspeed，threshold〉

其中 starttime：移动对象位于起始位置(x.startposition，y.startposition)时的时间。direction：移动对象的预计运动方向，用移动轨迹和 x-轴的夹角 θ 表示。speed：移动对象运动的估计速度。maxspeed：对象经过位置更新点时的最大速度。threshold：距离的阈值。任何时刻实际位置和计算位置的偏差应小于 o.threshold，当偏差大于等于 o.threshold 时就会引起位置更新操作[3]。

使用位置记录，不用频繁地更新移动对象的位置信息就可估算出其大概位置。

定义2 根据位置记录计算的移动对象的近似位置定义为数据库位置(database position)，时刻 tx 的数据库位置(x，y)表示为

$$\begin{cases} x=o.x.startposition+o.speed\times\sin\theta\times(tx-o.starttime) \\ y=o.y.startposition+o.speed\times\cos\theta\times(tx-o.starttime) \end{cases}$$

其中，o.starttime 为最近一次位置更新的时刻。

实际上，移动对象在位置更新后通常不会以匀速 o.speed 沿方向 o.direction 运动，因而数据库位置和实际位置之间会有偏差。为了限定移动对象位置的不确定性，当数据库位置和实际位置(ax，ay)之间的偏差大于等于预定义值 o.threshold 时，将执行位置更新。也就是说，在任何时刻 t，如果

$$deviation=\sqrt{(ax-x)^2+(ay-y)^2}\geqslant o.threshold$$

系统将更新对象 o 的位置信息，我们称之为基于距离的位置更新策略。

2.3 位置数据库组织

与个人通信系统中的层次位置数据库类似[4]，系统中的位置数据库也组织成树形结构，如图2所示。叶子位置数据库和一个 MSS 对应，它管理该移动单元中所有移动对象的位置信息。一个位置记录是一个定义1中的7元组，父结点位置数据库包含了它所有孩子结点位置数据库中移动对象的位置记录。

在个人通信系统的层次位置数据库中，内部位置数据库(inner location database)中移动对象位置信息有两种表示方法。一是指向孩子结点位置数据库的指针，移动对象的实际位置信息包含在孩子结点位置数据库中，我们称之为指针表示法(Pointer Paradigm，PP)。二是内部位置数据库中移动对象的位置信息就是实际位置信息，移动对象的位置信息在所有的位置数据库中都一样，我们称之为实际位置表示法(Actual Location Paradigm，ALP)。

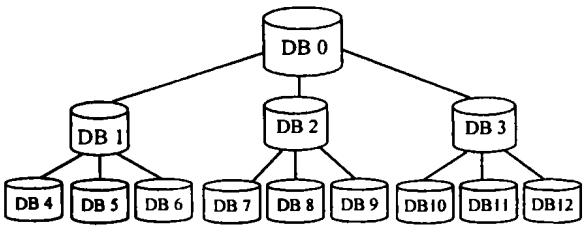

Figure 2 An example of hierarchical location databases

这两种表示法都不适用于连续的位置相关查询的应用。使用 ALP 时，因为系统中包含大量的移动对象，所以内部位置数据库的存储空间开销太大以至查询效率很低。使用 PP，内部数据库中的移动对象位置信息是指向孩子位置数据库的指针，实际位置信息只有在叶子位置数据库中才能获得，使得当 LDCQ 涉及到多个移动对象时很难确定 LDCQ 的处理节点。

在我们的系统中，所有位置数据库中移动对象的位置记录都有相同的格式，由7个属性组成。但在同一段时间内，不同位置数据库中同一移动对象的位置记录可能包含不同的内容。位置数据库层次越高，包含的位置信息越多，相对叶子位置数据库执行的位置更新操作也更多，这使得内部位置数据库成为影响系统性能的瓶颈，为了减少其中位置更新的数量，对同一个移动对象，叶子位置数据库位置记录的阈值应小于内部位置数据库位置记录的阈值。

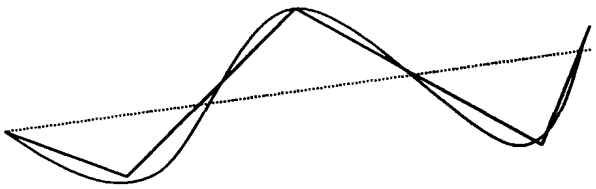

Figure 3 Comparison between the leaf location database and the inner location databases

设移动对象 o 沿着图3所示的曲线运动。叶子位置数据库中用直实线来模拟 o 的运动轨迹，内部位置数据库中则用虚线模拟对象的运动轨迹，因此叶子位置数据库的位置记录比内部位置数据库更能准确地描述移动对象的运动轨迹。

3 位置相关查询处理节点的确定

当连续的位置相关查询(从移动主机或者固定主机)提交给系统时，系统根据 LDCQ 的不同特性来选择处理节点。

(1)LDCQ 查询对象是静止对象的集合，例如移动车辆可能发出如下的 LDCQ 查询请求：

"查询距我5公里内所有的汽车旅馆，直到有一家在价钱和环境上都令人满意为止。"

我们称这种 LDCQ 查询为静止对象-移动对象(stationary object-moving object：SM)LDCQ，并假定静止对象在位置数据库中也有位置记录(记录内容是常数)。

(2)LDCQ 查询某些移动对象相对于另一移动对象的位置，例如移动车辆发出如下 LDCQ 查询请求：

"查询距我10公里内的所有移动车辆，直到有一个愿意提供帮助为止"

我们称这种查询为移动对象-移动对象(moving object-moving object，MM)LDCQ。MM-LDCQ 查询处理的节点应包含所有满足查询条件的移动对象，并尽可能不包含不满足查询条件的移动对象。

设 RO_{LDCQ} 为 LDCQ 中的参照对象(发出查询请求的对象)，对一个 LDCQ，每个位置数据库中的移动对象可分为三类：PMOS，UDMOS，IMPMOS。PMOS 类对象最可能满足 LDCQ 条件，UDMOS 类对象不一定满足条件，而 IMPMOS 类对象在将来一段时间内也不会满足条件。我们使用距离 dis1，dis2对移动对象进行划分，其中 dis1为 LCDQ 中的距离值，dis2为一预测值。系统计算出数据库中每个移动对象 o 与参照对象 ROLDCQ 之间的距离，记作 dis(ROLDCQ，o)。如果 dis(ROLDCQ，o)≤dis1，则对象 o 归为 PMOS，如果 dis1≤dis(ROLDCQ，o)≤dis2，则对象 o 归为 UDMOS，如果 dis2≤

dis(ROLDCQ,o),对象o归为IMPMOS。

下面给出确定LDCQ处理节点的算法:

步骤一:找出层次位置数据库中与ROLDCQ对应的叶子位置数据库,记为LDB。

步骤二:按上述方法计算移动对象集合:$PMOS_c$,$UDMOS_c$,$IMPMOS_c$。

步骤三:LDB=LDB.Parent

步骤四:计算移动对象集合:$PMOS_p$,$UDMOS_p$,$IMPMOS_p$。

步骤五:while ($PMOS_c$!= $PMOS_p$) OR ($UDMOS_c$!= $UDMOS_p$)

```
BEGIN
        PMOS_c=PMOS_p;
        UDMOS_c=UDMOS_p;
        IMPMOS_c=IMPMOS_p;
        LDB=LDB.Parent;
          计算移动对象集合:PMOS_p,UDMOS_p,
        IMPMOS_p
        END
```

完成上述操作后,选择LDB作为LDBQ的处理节点。

4 LDCQ处理方法

4.1 连续的位置相关查询的正确性标准

移动计算系统中,利用位置数据库的位置记录可获得移动对象在某一时刻的近似位置。由于移动对象一般不会按照位置数据库中设置的路线运动,所以会造成偏差,而结果的正确性也会受这种偏差的影响,但下面两种情形中的错误结果对移动客户来说非常严重,要尽量避免。

(1)移动对象的实际开始时间等于当前时间,而对应元组中的开始时间大于实际的开始时间。

(2)元组中开始时间小于移动对象的实际开始时间,且等于当前时间。

第一种情况中,对象目前满足LDCQ的查询条件,而系统却错误地认为对象在将来满足条件。第二种情况中,对象目前并不满足LDQ的查询条件,而系统却错误地认为对象目前满足条件。由于系统不可能知道所有移动对象在任何时刻的实际位置,所以这类错误很难完全避免。系统要实现的目标是使实际开始时间与元组中的开始时间的偏差最小,为了描述更准确,我们给出下面公式:

$$\eta=\frac{|actualbeg\ int\ ime-tuplebeg\ int\ ime|}{actualendtime-actualbeg\ int\ ime}$$

在LDCQ处理中,即使不能够准确地给出移动对象的实际开始时间,也应尽量使η最小化。

4.2 基于位置更新的LDCQ处理方法及其相关问题

位置相关查询的结果是元组〈O,begin,end〉集合,如果查询是连续的,则当LDCQ所涉及的对象有位置更新时,将从新执行查询,并将新的查询结果传给发出LDCQ请求的移动(固定)主机。这种基于位置更新的LDCQ处理方法需要大量的CPU时间,同时无线带宽开销也很大。

例如一个移动车辆可能发出如下的连续位置相关查询的请求:

"查询所有距我3公里内的汽车旅店,直到有一个满足我条件的为止。"

系统根据移动对象目前的位置、预计速度和方向来处理LDCQ,查询结果是元组〈M,begin,end〉的集合,其中M表示从时间begin到end距移动对象3公里内的旅店。设LDCQ提交给系统的时刻是0,M1在时刻4到9满足条件,在结果集中有元组:

t=0:〈M1,4,9〉

如果移动对象在时刻2有位置更新,系统将重新处理LDCQ,其结果集中有元组:

t=2:〈M1,6,10〉

如果移动对象在时刻3又有位置更新,系统又重新处理LDCQ,其结果集中有元组:

t=3:〈M1,5,11〉

从上面分析可以看出,既然在时刻3M1仍不能满足LDCQ查询条件,且时刻3元组的开始时间和结束时间几乎和时刻0的元组一样,因此没有必要处理时刻2的LDCQ,也不需要传给移动对象。因此,需要对LDCQ处理方法进行改进以减小CPU处理代价和无线带宽的开销。

4.3 基于对象最大速的延迟的LDCQ处理方法

设t为最近一次LDCQ查询的时刻,t_u表示移动对象下一次位置更新的时刻,在下面两种情况下,没必要重新处理LDCQ。

(1)t<begin,且移动对象在时刻t_u仍不满足LDCQ条件(设对象以最大速度运动)。也就是说移动对象不仅在时刻t不能满足条件,且在下一次位置更新时也不能满足查询条件。

(2)t=begin,且移动对象在时刻t_u一定能满足LDCQ查询条件(设对象以最大速度运动)。也就是说从最近一次执行LDCQ的时刻到下次位置更新移动对象始终满足查询条件。

下面问题是怎样确定LDCQ中涉及的对象能够一直满足(不满足)查询条件。在延迟的LDCQ处理方法中,定义引发重新执行LDCQ的条件如下:

(1)设t_L为最近一次执行LDCQ的时刻,元组〈O,begin,end〉中O表示从时间begin到end满足条件的对象。Min-Time(O)表示对象以最大速度运动时不再满足LDCQ条件的最小时刻。

(2)在时刻Min-Time(O),如果在LDCQ处理之后对象O或参照对象至少有一次位置更新,那么LDCQ查询要重新处理,Min-Time(O)也要重新计算。否则LDCQ处理延迟到对象O或参照对象位置更新时进行。

延迟的LDCQ处理方法的主要思想是尽量延迟对象O的LDCQ处理,下面两节将详细讨论两种不同的LDCQ处理方法中Min-Time的计算方法。

4.3.1 SM-LDCQ中的Min-Time计算方法 执行SM-LDCQ查询时,系统维护所有静态对象到移动对象的距离信息。SM-LDCQ查询结果内部表示为包含四个属性的元组集合,形式如下:

〈SOID,ad,timer,ntime〉

其中SOID表示静止对象,ad表示某次执行SM-LDCQ时移动对象与静止对象的实际距离。timer是一个计数器,用来计数从执行SM-LDCQ时刻到现在的时间。当实际距离小于SM-LDCQ中的距离时,该对象满足SM-LDCQ查询条件,由于移动对象以一定的速度运行,所以在一段时间T内静止对象都满足SM-LDCQ中的查询条件。

设md表示SM-LDCQ中的查询距离,mo表示SM-LDCQ中涉及的移动对象,mo.maxspeed表示mo的最大速度,mo在时间t内运动的最大距离为:mo.maxspeed×t。

如图4所示，ad 表示最近一次 SM-LDCQ 查询过程中静态对象到移动对象 o 之间的距离。在时间 t 内，移动对象运动的最大距离是 maxspeed×t，移动对象 mo 与静止对象间距离 ada 满足下列不等式：

$$|ad-mo.\max speed\times t|\leqslant ada\leqslant ad+mo.\max speed\times t$$

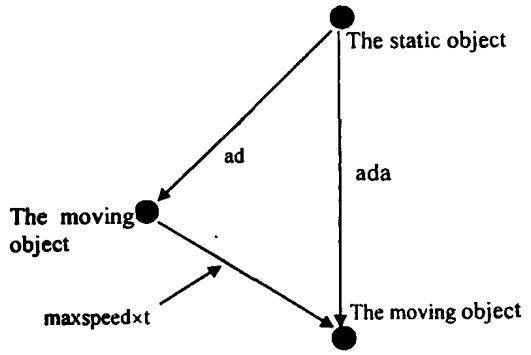

Figure 4 The maximum distance of between the objects after time t

由上面分析可得在时间 $|md-ad|/mo.\max speed$ 内移动对象 mo 一定满足 LDCQ 查询条件。对4元组〈SOID，ad，timer，ntime〉，令

$$ntime=|md-ad|/mo.\max speed$$

timer 的初值设为 t_L（对 SOID 进行 LDCQ 处理的时间点），其值随着时间而增加，当 timer 值等于 t_L+ntime 时，系统会按照4.3节的步骤(2)执行。

4.3.2 MM-LDCQ 处理中的 Min-Time 的计算方法

MM-LDCQ 功能是查询相对于移动被参照(referenced)对象的所有移动参照(referencing)对象的位置。设 ad 为某次执行 MM-LDCQ 时移动参照对象 rio 与移动被参照对象 rdo 间的距离，我们给出下列定义：

rio.maxspeed： 移动对象 rio 的最大速度

rdo.maxspeed： 移动对象 rdo 的最大速度

md： MM-LDCQ 中的查询距离

完成 MM-LDCQ 处理后 rio 与 rdo 间的距离应满足下面的不等式：

$$|ad-(rio.\max speed+rdo.\max speed)\times t|\leqslant ada\leqslant ad+(rio.\max speed+rdo.\max speed)\times t$$

在执行完 MM-LDCQ 后的时间 $|md-ad|/(rio.\max speed+rdo.\max speed)$ 内，对象 rio 一定满足 LDCQ 查询条件，所以当对象 rio 和 rdo 的位置更新时，不必重新执行 MM-LDCQ 操作。

MM-LDCQ 的查询结果内部也表示为4元组〈MOID，ad，timer，ntime〉。其中 MOID 表示移动对象，timer 是一个计数器，用以计数从最近一次对 MOID 处理完成后开始的时间，LDCQ 处理完后将 timer 值设为 t_L，timer 的值随时间增加。

$ntime=|md-ad|/(rio.\max speed+rdo.\max speed)$ 表示移动对象 MOID 能够连续满足 MM-LDCQ 条件的最小时间段。当 timer 等于 t_L+ntime 时，系统执行4.3节的步骤(2)。

结束语 为实现连续的位置相关的查询，移动计算系统要密切地跟踪移动对象的位置。使用传统的位置数据库组织模型和数据库技术，移动对象位置的频繁更新会造成无线带宽巨大的开销。本文在 MOST 数据模型的基础上提出了一种新的位置数据库组织模型，用以对移动对象动态位置建模。在该模型的基础上，我们还提出了两种 LDCQ 处理方法，以其实现减小 LDCQ 处理的 CPU 代价和无线带宽的开销。

目前，连续的位置相关查询的处理技术还远不成熟，有很多问题亟待解决。例如，对于 MM-LDCQ，被查询的对象和提交查询请求的对象都在移动，这两类移动对象间距离阈值的分布对系统性能有重要影响，这将是我们下一步要解决的问题。

参考文献

1 Wolfson O, Xu B, Chamberlain S, Jiang L. Moving objects databases: issues and solutions. In: Proc. of the 10th Intl. Conf. on Scientific and Statistical Database Management (SSDBM98), Capri, Italy, July 1998. 111～122

2 Wolfson O, et al. Cost and imprecision in modeling the position of moving objects. In: Proc. of the Fourteenth Intl. Conf. on Data Engineering (ICDE14), Orlando, FL, Feb. 1998

3 Wolfson O, et al. Updating and querying databases that track mobile units, invited paper, Special issue of the distributed and paralle databases Journal on Mobile Data Management and Applications, Kluwer Academic Publishers, 1999, 7(3): 257～287

4 Pitoura E, Samaras G. Locating objects in mobile computing. IEEE Transaction on Knowledge and Data Engineering. Accepted, 2000. To appear

5 Gok H G, Ulusoy O. Transmission Of Continuous Query Results In Mobile Computing Systems. Information Sciences, 2000, 125 (1-4): 37～63

6 Gök H G. Processing of continuous queries from moving objects in mobile computing systems: [PhD thesis]. Department of computer engineering and information science, Bilkent University, Jan. 1999

(上接第81页)

读的信息用用户同口令一起传来的密钥加密传给用户。这里用户可根据需要，定期更改口令。更改口令时，只需要把旧口令和新口令一起加密传给 NASS，NASS 就在用户口令登记表中用新口令替代旧口令。

5 初步实验结果

我们分别对 NASS 不加载安全模块的情况及加载安全模块的情况，分别测试其速度，其结果示于图5。从显示的结果看，加载安全模块写比不加载安全模块写慢7%～12%。加安全模块的读比不加安全模块的读慢11%～19%。

参考文献

1 Brocade whitepaper: comparing the Storage Area Network and Network Attached Storage

2 HITACHI whitepaper: Planning for Network Attached Storage and Storage Area Network Convergence

3 Blaze M. Key Management in an Encrypting File System. In: Proc. of the Summer 1994 USENIX Conf. 1994

4 Gobioff H. Security for a High Performance Commodity Storage Subsystem: [Ph. D. thesis]. Computer Science Department. Carnegie Mellon University, July 1999. Available as Technical Report CUS-CS-99-160

5 Krawczyk H, Bellare M, Canetti R. HMAC: Keyed-Hashing for Message Authentication. IETF Network working Group RFC2104, Feb. 1997

6 Reed B, Chron E, Burus R, Long D. Authenticating Network Attached Storage. IEEE Micro, 2000, 20(1): 49～57