第 33 卷 第 1 期 计 算 机 工 程 2007 年 1 月
Vol.33 No.1 Computer Engineering January 2007
·软件技术与数据库· 文章编号：1000—3428(2007)01—0047—03 文献标识码：A 中图分类号：TP391

一种基于移动 Agent 的查询处理策略

黄 艳，朱会东
(郑州轻工业学院计算机与通信学院，郑州 450002)

摘 要：介绍了移动 Agent 技术的特点，查询包含与归并的思想，并在此基础上提出了基于移动 Agent 的查询处理策略。通过合并查询减少了查询处理量；而使用移动 Agent 来处理查询请求与查询结果减少了网络通信量。
关键词：移动代理；查询包含；查询归并；MSS

Query Transaction Strategy Based on Mobile Agent

HUANG Yan, ZHU Huidong
(Department of Computer and Communication Engineering, Zhenzhou University of Light Industry, Zhenzhou 450002)

【Abstract】This paper first narrates the characteristic of mobile agent and the rules of query containment and merging, then, based on it, proposes a query transaction strategy. By incorporating queries, the amount of query becomes lower, and by using mobile agent, the network communicating amount decreases.
【Key words】Mobile agent; Query containment; Query merging; Mobile support station

近年来随着计算机网络的日益发展，人们对通过网络获取信息的依赖性越来越强，这不仅体现在获取和提交信息量的增大，更体现在对获取信息的实时性和便利性的迫切需求上。在 20 世纪 90 年代初期，功能强大的便携式计算机及无线局域网的出现迫使学者研究这样一个问题：如何建立带有移动计算机的分布式计算系统，使得一个便携式计算机可以通过无线通信经由基站访问有线网络上的数据信息。因此，移动计算(mobile computing)应运而生。移动通信和网络技术的高速发展，是推动移动计算技术走向实际应用的重要力量。移动通信网络构成了移动计算环境的物理通信基础。

1 移动 Agent

Agent 技术是面向对象技术向软件智能化发展的产物，它的出现将计算机软件设计提高到一个更高的抽象层次。人们可以把它看作一个自治的实体，它能够感知环境，并且对外界的信息作出一定的判断与推理，来控制自己的决策与行为，以便完成一定的任务。Agent 技术和分布式对象技术相结合产生一个新的研究领域——移动 Agent，该技术的目的就是如何简化分布计算的任务复杂性。

移动 Agent 实质上是一个封装代码、运行状态和数据的计算实体，是可以在执行过程中，有目的地、自治的在网络中移动，利用与分布资源的局部交互而完成分布任务的软件实体。其移动是主动的，移动 Agent 模式下没有客户机和服务器之分，或者说是传统的 C/S 分布计算的扩展方式，称为远程程序设计(RP)。在远程程序设计交互模式中，移动 Agent 具有感知网络状态的功能，如网络连接是否正常，网络的当前载荷等，以适应网络的配置和变化，在反映规划的控制之下正确地驱动。移动 Agent 可以感知软件资源的条件，如资源是否可以利用，数据库中的数据的特殊变化等，对重大事件作出反应。移动 Agent 具有自治的决策能力，由于信息是息息相关的，利用对所访问的网络接点的信息反馈，移动 Agent 可以独立的修改整体规划。Agent 的运行环境统称为移动环境，移动 Agent 可以根据自己的需要从一台主机上移动到另一台主机上，而且可以进行多次移动，这完全取决于其本身。

移动 Agent 是一种新的网络计算技术，它能有效地降低分布式计算中的网络负载、提高通信效率、动态适应变化了的网络环境，并具有很好的安全性和容错能力。为有效地进行数据库访问提供了一种新思路和新方法。移动 Agent 目前已经从理论探索进入到实用阶段，涌现出了一系列较为成熟的开发平台和执行环境。

2 基于移动 Agent 查询处理策略

服务器数据分发和客户端缓存管理这两个研究论题都是特别针对移动数据库系统中的移动用户而言的，并没有把移动计算环境作为一个整体来考虑。基于传统的分布式环境下的查询研究虽然在移动计算环境下仍然可行，但却没有考虑到随着越来越多的移动用户的加入及固定网络的不断扩充，使得查询处理受到极大影响。尽管计算机网络技术也在不断发展，网络环境有一定程度的改善，但是通过改善查询处理从而改善网络状况往往能有事半功倍的效果。

随着网络规模的不断扩大，网络数据一般都是分布存储在网络中各个结点上，而对这些数据的处理一般都是在网络中设置有专门的服务器来处理。对于数据查询一般都是设置查询服务器来处理。本文对于移动查询的讨论就是基于此种模式，即移动环境下，所有用户查询都交由查询服务器来处理。查询服务器可以是一个或多个网内站点。

2.1 查询请求调度模型

查询服务器一方面要处理固定网络中一般用户的查询请求，另一方面还要处理其管辖区内的移动用户的查询请求，

作者简介：黄 艳(1977－)，女，硕士、助教，主研方向：分布式数据库技术；朱会东，硕士
收稿日期：2006-01-12 **E-mail：**h_y2002@sohu.com

为了公平有效地处理这些查询请求，本文给出了一种用户查询请求调度模型，如图 1 所示。

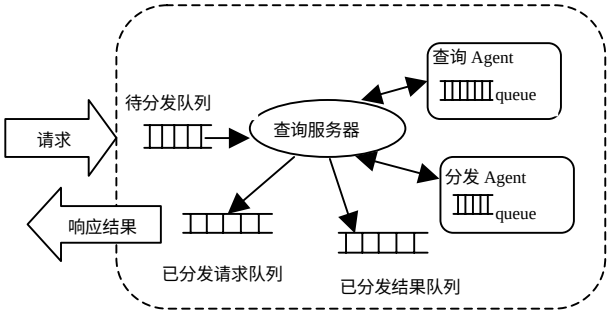

图 1　查询调度模型

其中，查询请求之间相互独立；查询服务器采用先来先服务原则分发请求，其待分发队列存储尚未分发给查询 Agent 的请求，已分发请求队列存储等待响应结果的已分发请求，主机采用时间片轮转法为驻留在该机上的查询 Agent 提供 CPU 资源，以处理请求等待队列 queue 中的请求；已分发结果队列中存储已经由查询 Agent 取得查询结果，分发 Agent 正在往其目的地发送的查询请求。

(1)待分发队列

查询服务器设置有一个收信箱，收信箱维护一个先进先出队列，为每一个到达的查询请求建立一个元素，顺序保存在队列中。查询服务器将为收信箱该队列中每个用户查询请求提供查询服务。

(2)已分发请求队列

查询服务器设置有一个发信箱，发信箱维护一个先进先出队列，用来保存等待响应结果的已分发请求。查询服务器正在为发信箱该队列中每个用户查询请求提供查询服务。

(3)已分发结果队列

查询服务器设置有一个发信箱，发信箱维护一个先进先出队列，用来保存等待发送结果的已分发请求。查询服务器正在为发信箱该队列中每个用户查询请求提供发送结果服务。

2.2 查询包含与归并

对于同一数据集，大量用户将产生许多相似的查询事务，这必将造成有线网络负载的增加从而可能导致网络的拥塞和某些结点的瓶颈。本文中查询服务器利用用户对于同一数据集查询请求之间的一些关联关系，对待分发队列先进行查询的包含与归并处理来优化查询请求，减少查询请求数量，再由查询 Agent 来处理合并后的查询请求。

2.2.1 查询包含

设火车车票信息存放在关系数据库 R(start, time, price, ID)中，其中 start、time、price 表示火车从源站出发时间、到目的站路途持续时间和票价，ID 表示车次。假设有两客户，A 要查询当天从郑州到北京路途持续时间不超过 10 个小时的车次信息，B 要查询当天从郑州到北京路途持续时间不超过 8 个小时的车次信息，很显然，A 的查询结果将包含 B 的查询结果，这就是简单的查询包含。

定义 1　假如查询Q_i的结果集是查询Q_j的结果集的一个子集，则称查询Q_i被查询Q_j包含，表示为$Q_i \subseteq Q_j$。

2.2.2 查询归并

对上述关系数据库 R，现在客户 A 要查询当天晚上 8 点到 11 点出发、路途持续时间不超过 10 小时、票价不超过 100 元的从郑州到北京的车次信息，如图 2 中的粗线条立方体所示；移动客户 B 要查询当天晚上 6 点到 10 点出发、路途持续时间不超过 8 小时、票价不超过 120 元的从郑州到北京的车次信息，如图 2 中的细线条立方体所示。那么可以把这两个查询归并成一个：当天晚上 6 点到 11 点出发、路途持续时间不超过 10 小时、票价不超过 120 元的从郑州到北京的车次信息，如图 2 中的虚线立方体所示。这就是查询归并的思想。通过归并可以把若干个相似查询归并成单一的查询。

定义 2　设Q_i，$1 \le i \le m$表示参与查询的m个客户提出的查询请求；$S_Q=\{Q_1,Q_2,\ldots,Q_m\}$表示客户提出的$m$个查询请求的集合；定义$M(S_Q)=\{M_1,M_2,\ldots,M_n\}$为对查询集合$S_Q$应用查询归并算法所得的结果，其中$M_j(1 \le j \le n)$属于集合$S_Q$的一个子集，即一个相似查询集合；定义$MERGE(M_j)$，$1 \le j \le n$，为对集合$M_j$中查询进行归并后所得的新查询。

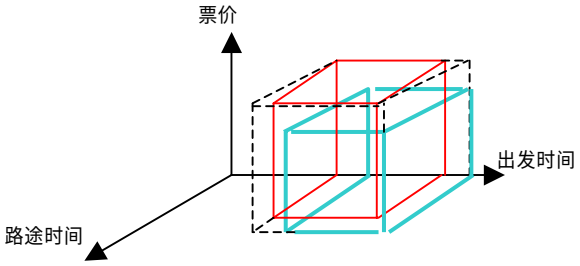

图 2　查询归并示例图

2.3 查询处理算法中主要数据结构定义

(1)查询消息逻辑结构

查询服务器为每个到达的用户请求生成固定的查询消息，并存储在收信箱中。查询消息表 Query_original(Msg_id，Rev_time，Query_msg，Address)各字段的含义如下：

Msg_id：查询请求顺序号，由系统自动生成；

Rev_time：收到查询请求的时间；

Query_msg：查询请求内容；

Address：发出查询请求的用户网络地址。

(2)合并消息队列

查询服务器设置有一个合并信箱，用来保存利用基于包含与归并的查询优化算法处理过的查询请求消息。

(3)合并消息逻辑结构

查询服务器为利用算法合并的查询项生成合并消息，存储在合并信箱中。合并消息 Query_unite(Msg_ids，Merge_id，Merge_msg，Msg_unite，Merge_type) 各字段的含义如下：

Msg_ids：合并后查询所包含的合并前查询请求顺序号集；

Merge_id：合并后查询请求号，由算法生成；

Merge_msg：合并后查询请求消息内容；

Msg_unite：合并后的查询集合；

Merge_type：合并操作类型(contain / merge)。

2.4 查询处理策略

2.4.1 查询请求处理

假如服务器短时间内收到原始查询请求集合$S_1=\{Q_1,Q_2,\ldots,Q_{20}\}$，服务器把查询请求集合$S_1$中的查询请求信息存储到表Query_original中，然后，查询服务器对查询集合S_1的处理过程如下：

(1)把Query_original表中查询请求集合S_1所有具有包含关系的查询请求合并，得到新的查询请求集合$S_1'=\{Q_1',Q_2',$

Q_3', Q_7, Q_8, Q_9, Q_{10}}，其中对于任一Q_i' $\subseteq$ S_Q'，都有Q_i'是S_Q的一个子集，如Q_1'＝{Q_1, Q_2, Q_3}。

(2)在Query_unite表中增加包含合并前后查询请求对应关系的记录项，如((Q_2, Q_1, Q_3)，Q_1'，(略)，contain)。这里，为了方便后面对查询结果集的处理，在包含合并过程中对Query_unite表中Msg_ids信息项的查询请求顺序号按查询结果集从大到小排序，如记录项((Q_1, Q_2, Q_3)，Q_1'，select语句，contain)经排序后为((Q_2, Q_1, Q_3)，Q_1'，(略)，contain)。

(3)检查合并后的查询请求量，如果合并后的查询请求量已经达到了服务器处理能力的范围，则查询合并结束；否则，还需要做两两查询归并处理，直到满足服务器处理要求。

(4)两两归并的过程中，在查询对应关系表Query_unite中记录下归并前与归并后查询请求的对应关系；得到归并后的查询请求Q_1''，Q_2''。

(5)得到最终的表 Query_unite 如表 1 所示。

表 1　Query_unite

Msg_ids	Merge_id	Msg_unite	Merge_msg (略)	Merge_type
(Q_2,Q_1,Q_3)	Q_1'	Q_1', Q_4, Q_5,Q_6, Q_7, Q_8, Q_9, Q_{10},Q_{11},Q_{12}, Q_{13}, Q_{14}, Q_{15}, Q_{16},Q_{17}, Q_{18}, Q_{19} , Q_{20}		contain
(Q_4,Q_{11},Q_6, Q_5,Q_{12}, Q_{13})	Q_2'	Q_1',Q_2', Q_7, Q_8, Q_9, Q_{10}, Q_{14}, Q_{15},Q_{16},Q_{17}, Q_{18}, Q_{19}, Q_{20}		contain
(Q_{18},Q_{20},Q_{16},Q_{17}, Q_{14},Q_{19},Q_{15})	Q_3'	Q_1',Q_2', Q_3', Q_7, Q_8, Q_9, Q_{10}		contain
(Q_2', Q_3', Q_9)	Q_1''	Q_1',Q_1'', Q_7, Q_8, Q_{10}		merge
(Q_1', Q_{10})	Q_2''	Q_2'',Q_1'', Q_7, Q_8		merge

2.4.2 查询结果处理

接下来查询服务器对由表Query_unite中得到的新查询集合Msg_unite={Q_2'',Q_1'' , Q_7, Q_8 }中的每个查询请求分别分派查询Agent去收集查询结果。假设查询Agent对查询Q_1''收集得到查询结果R_1''，为了减少网络传输流量，查询服务器R_1''的处理过程如下：

(1)查表Query_unite知Q_1''＝ (Q_2', Q_3', Q_9)，据表Query_original把查询结果R_1''直接发送给Q_9请求用户；

(2)在R_1''结果集上运行Q_2', Q_3'语句，分别得到查询结果子集R_2'，R_3'；

(3)继续查表Query_unite知Q_2'=(Q_4, Q_{11}, Q_6, Q_5, Q_{12}, Q_{13})，Q_3'=(Q_{18}, Q_{20}, Q_{16}, Q_{17}, Q_{14}, Q_{19}, Q_{15})；

(4)对于查询结果子集R_2'，由查询服务器派生出分发移动Agent(R_2')，Agent(R_2')将沿着(Q_4→Q_{11} →Q_6→Q_5→Q_{12}→Q_{13})的迁移路径依次把查询结果发送到目的地。由于(Q_4, Q_{11}, Q_6, Q_5, Q_{12}, Q_{13})中各查询请求是按查询结果集从大到小排序的，因此，在迁移的过程中，Agent(R_2')每经过一个站点，都要执行其相邻下一站点的查询请求语句，以得到新的逐渐减少的查询结果集，如Agent(R_2')在经过Q_4所在站点时，先把查询结果R_2'传送给它，接着执行Q_{11}的查询语句得到新的减小的结果集R_{11}，然后携带R_{11}往其所在地迁移，以此类推，直到迁移到最后一个站点——Q_{13}所在站点，此时Agent(R_2')携带最小的结果集R_{13}。

(5)对每一个包含合并查询的结果集都重复(4)，直到每一个查询站点都得到结果。

以上处理过程中，对于每一个包含合并查询请求集合S_i={Q_j,Q_k,…,Q_m}(Q_j,Q_k,…,Q_m之间是两两有包含关系的查询请求,且按结果集大小排序)，每个查询请求结果集都将包含排在其后的所有查询请求结果集集合，如Q_j查询请求结果集将包含Q_k,…,Q_m的查询请求结果集集合。并且对于S_i，由查询Agent收集得到的查询结果集R_i正好对应于Q_j的查询结果集。这样，分发Agent按着以上策略在沿着集合S_i中查询请求站点顺序发送结果集时，沿着该路线上的网络流量是逐渐地减少的。如最初到达Q_j站点的查询结果数据量可能达到几十兆字节，经由Q_k,..., Q_{m-1}，最后到达Q_m站点的查询结果数据量可能只有几千字节。这种方法和以往常用的传统方法相比较能使得网络流量大大减少，极大地节约了网络带宽，从而使网络负担减轻，加快网络数据传输速率。

3 性能实验与结果分析

为了研究算法性能，作者对原来的自动查票系统稍加改进。原来的自动查票系统是用多线程实现的，系统一边把收到的查询请求放入收信箱中，一边从收信箱中顺序取出查询信息交由查询 Agent 进行处理。由于对数据库表操作的互斥性，在处理查询请求前设置了临界区，因此查询请求只能一个个串行执行。原系统对于查询结果的分发采用的是传统的一(服务器)对一(客户端)发送方式。在改进的自动查票系统中，一旦收信箱中的未处理查询请求数超过某一限值，就自动执行查询的包含与归并对查询请求进行合并，查询服务器只派生查询 Agent 对合并后的查询进行处理，得到的查询结果由分发 Agent 按上述策略进行分发。

由于实验环境的限制，实验中没有考虑到移动用户。在实际应用中移动用户由 MSS(支持移动计算的固定站点)来管理，因此可以把 MSS 看作用户和一般用户一起统一处理。

实验时考虑到查询的随机性，少数查询请求之间的交集有限，所以取限定值 30，即一旦收信箱中的未处理查询请求数等于或超过 30，就自动进行查询的包含与归并处理。表 2 是原系统与改进系统实验结果数据。

表 2 算法性能测试数据与原系统测试数据比较

收件箱查询请求数	原系统查询处理时间(s)	改进系统的查询处理时间		
		查询请求处理时间(s)	查询结果处理时间(s)	总处理时间(s)
30	92	44	35.1	79.1
60	177	56.1	67	123.1
90	280	55.2	100	155.2
120	352	62	113.2	175.2
150	411	60.4	133	193.4
180	509	63.2	143.2	206.4
210	622	51.1	160.5	211.6
240	690	66	153.7	219.7
300	868	68.3	158.6	226.9

由上表可以计算出系统改进前后的单位查询请求平均处理时间，如图 3 所示。

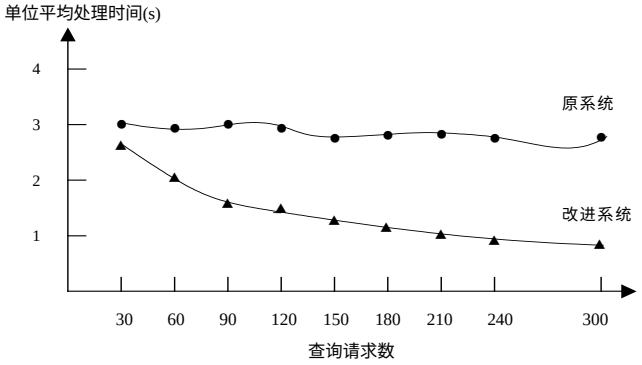

图 3 单位查询请求平均处理时间比较

(下转第 52 页)