On-Demand数据广播环境下实时有序查询处理

王洪亚[1] 刘晓强[1] 何浩源[1] 宋 晖[1] 肖迎元[2] 乐嘉锦[1]
(东华大学计算机科学与技术学院 上海 201620)[1]
(天津理工大学计算机科学与技术学院 天津 300191)[2]

摘 要 在On-Demand数据广播环境下,广播服务器基于用户发送的数据请求等信息进行调度决策来满足用户的数据访问需求。在很多实际应用中,用户的数据请求需要在一定时间段内得到满足,即数据请求是有截止期的。现有研究只考虑了具有截止期约束的单个数据请求的调度问题,而实时查询处理即用户以查询为单位依次发送多个数据请求的研究尚未得到足够的关注。本文重点研究了On-Demand数据广播环境下如何有效地处理实时有序查询这一问题。基于对该问题的分析,定义了一类新的调度问题ROBS并证明了ROBS的Off-Line版本是NP-Hard的;提出了一种新的考虑查询语义的On-Line调度算法OL-ROBS,该算法通过综合考虑数据请求个数、查询截止期和查询剩余数据请求个数来确定待广播数据项的优先级;为提高OL-ROBS的执行效率,设计了一种裁减算法,用以减少调度决策的搜索空间。模拟实验将OL-ROBS与目前最为有效的实时数据请求调度算法Sin-θ进行了比较,结果显示OL-ROBS具有更低的错过截止期比率。

关键词 数据广播,实时有序查询处理,调度算法
中图法分类号 TP31 **文献标识码** A

Real-time Ordered Query Processing in On-Demand Broadcast Environments

WANG Hong-ya[1] LIU Xiao-qiang[1] HE Hao-yuan[1] SONG Hui[1] XIAO Ying-yuan[2] LE Jia-jin[1]
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)[1]
(School of Computer Science and Technology, Tianjin University of Technology, Tianjin 300191, China)[2]

Abstract Existing research efforts on real-time data dissemination in on-demand data broadcast environments are only concerned with scheduling single data request with deadline constraints. The issue of processing real-time ordered query in on-demand broadcast systems was investigated. Particularly, we first formally defined a new kind of scheduling problem called ROBS by formulating the real-time ordered query processing problem. We also showed that the ROBS problem is NP-hard. Secondly, a novel scheduling algorithm called OL-ROBS was proposed to address the on-line version of ROBS. To tackle the performance issue of OL-ROBS, a delicate data structure for managing data requests was constructed and an efficient pruning algorithm was proposed. Finally, extensive simulations were conducted and the empirical results show that OL-ROBS owns significant performance gain over the well-known Sin-θ algorithm.

Keywords Data broadcast, Real-time ordered query processing, Scheduling algorithms

1 引言

移动计算环境下的数据分发(Data Dissemination)研究旨在解决如何高效地将数据发送到移动主机上。近年的研究表明,数据广播是一种有效的数据分发技术[8,11-13]。根据广播模式可将数据广播分为推(Push)和按需(On-Demand)两种,其中推指广播服务器不考虑用户的数据需求而直接将数据组织成广播形式发送出去;按需指客户机显式地将数据请求发送到服务器,服务器按照一定的调度策略选择数据进行广播。文献[1,5]研究表明,在数据量大、用户数量和访问模式经常变化的场合,On-Demand数据广播模式具有更高的带宽利用率和可扩展性。

实时数据广播是考虑时间约束的数据广播技术[3,4,7,8,10]。数据广播中的实时约束主要来自两个方面:一是很多数据本身是有截止期的,超过了这个时间段数据将没有任何意义;二是许多用户希望在一个合理的时间段内能够获得自己请求的数据。文献[3]通过对比Round-Robin,FCFS,EDF等几种经典调度算法,发现EDF的综合表现最好;文献[4]提出综合考虑数据请求截止期和数据请求个数的Sin-θ算法;文献[7]研究了当数据项不是等大小情况下的广

到稿日期:2009-07-01 返修日期:2009-09-10 本文受国家自然科学基金(60903160),上海市科技攻关项目(06dz150003)资助。
王洪亚(1976-),男,副教授,主要研究方向为移动计算、实时计算和数据管理等,E-mail:hywang@dhu.edu.cn;刘晓强(1968-),女,副教授,主要研究方向为智能信息系统等;何浩源(1982-),男,硕士生,主要研究方向为移动数据管理;宋 晖(1971-),女,副教授,主要研究方向为数据挖掘;肖迎元(1973-),男,副教授,主要研究方向为实时、空间和移动数据管理等;乐嘉锦(1955-),男,教授,主要研究方向为数据库理论和实现等。

播调度算法；文献[8]对 On-Demand 数据广播环境下的实时事务处理进行了研究；文献[10]研究了在给定有截止期约束的数据请求下如何确定广播频段数量问题。

虽然在 On-Demand 数据广播环境下，具有截止期约束的单个数据请求的调度问题已经得到不少研究，但支持实时查询处理(用户以查询为单位依次发送多个数据请求)的数据广播技术研究尚未得到足够关注。在许多实际应用中(如 Web 访问等)，用户的数据请求通常以查询的形式出现，即用户依次请求语义相关的多个数据项[15]。针对这类应用，本文重点研究了 On-Demand 数据广播环境下的实时有序查询处理问题。基于对实时有序查询处理需求的分析，定义了一类新的调度问题 ROBS 并证明了 ROBS 的 Off-Line 版本是 NP-Hard 的；提出了一种新的考虑查询语义的 On-Line 调度算法 OL-ROBS (On-Line Real-Time Ordered Broadcast Schedule)，该算法综合考虑了数据请求个数、查询截止期和查询剩余数据请求个数来确定待广播数据项的优先级；为提高 OL-ROBS 的执行效率，设计了一种裁减算法用以减少调度决策的搜索空间。通过模拟实验将 OL-ROBS 与目前最为有效的实时数据请求调度算法 Sin-θ 进行了比较，结果显示 OL-ROBS 有更低的错过截止期比率，在最好情况下性能提高了近 30%。

本文第 2 节给出了系统模型和 ROBS 调度问题的定义；第 3 节详细介绍了 OL-ROBS 调度算法的原理及实现方法；第 4 节是模拟实验；第 5 节介绍了相关的工作；最后对全文做了总结和展望。

2 系统模型与 ROBS 调度问题

2.1 系统模型

On-Demand 数据广播的系统模型如图 1 所示，由客户端和服务器端两部分组成。

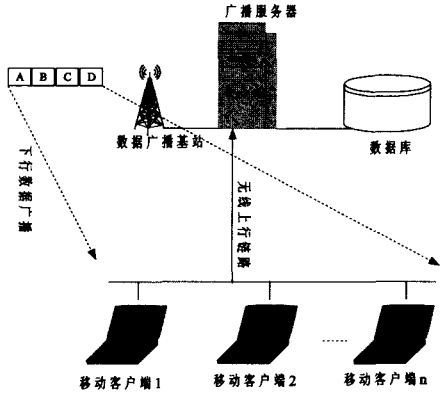

图 1 系统模型

客户端包括若干移动客户机，每个客户机独立发起实时有序查询(Real-Time Ordered Query)。本文假定同一时刻一个客户机只发起一个查询，且在当前查询执行完毕后再发起新的查询。q_i 表示第 i 个客户机发起的查询，$1\leqslant i\leqslant n$，$|q_i|$ 表示查询 q_i 读取数据项的个数 ($|q_i|\geqslant 1$)。查询通过上行链路发送数据请求到 MSS(Mobile Support Station)上，当且仅当 q_i 从数据广播中获取当前请求的数据项后才发起下一个数据请求。考虑数据广播环境下查询的高度可预测性，假定通过预分析技术在查询执行前可得到查询访问数据项的个数 $|q_i|$[9,15]。每个查询 q_i 被赋予一个截止期，若 q_i 在截止期后还未获得所需全部数据项，称 q_i 错过了截止期。本文采用实时计算领域广泛使用的固截止期模型[4,7](Firm Deadline Model)，即继续执行错过截止期的应用没有价值而应立即丢弃。若广播调度无法满足所有实时查询在截止期前执行完毕，则应尽可能减少错过截止期查询的比率，以提高系统的性能。

服务器端包括数据广播基站、广播服务器和数据库 3 部分。广播服务器负责管理上行和下行链路，通过上行链路接收查询发送的数据请求，执行给定的调度算法，生成广播调度程序。数据广播基站按照广播服务器的广播调度将数据项通过下行链路广播给客户机。数据库包括用于广播发送的数据。本文假定所有数据项的大小相同。

2.2 ROBS 调度问题

假定数据库为 N 个有限数据项组成的集合，即 $DB=\{d_m\mid 1\leqslant m\leqslant N\}$。

定义 1 查询 q_i 的第 j 个数据请求用 r_i^j 表示，每个 r_i^j 请求 DB 中的一个数据项 d_m。q_i 可形式化表示为两元组 $\langle r_i^1 r_i^2\cdots r_i^{|q_i|}, D_i\rangle$，其中 $r_i^1 r_i^2\cdots r_i^{|q_i|}$ 为有序数据请求序列，D_i 为 q_i 的截止期。

定义 2 $S^*=s_1 s_2\cdots s_{|S^*|}$，$s_k\in DB$，$1\leqslant k\leqslant |S^*|$ 为定义在 DB 上的有序数据项序列，其长度记作 $|S^*|$。对广播服务器而言，S^* 代表一个广播调度，称调度中每个数据项占用的位置为一个时槽(Slot)。

定义 3 当数据请求 r_i^j 发出后，若其所需数据项第一次在广播调度 S^* 中出现的时槽为 k，称 s_k 满足了 r_i^j，记作 $s_k\rightarrow r_i^j$。若 s_k 在时刻 D 之前被发送，称 s_k 在时刻 D 上满足 r_i^j，记作 $s_k\xrightarrow{D} r_i^j$。

定义 4 对查询 q_i，若广播调度 S^* 中存在一子序列 $s_{k_1}s_{k_2}\cdots s_{k_{|q_i|}}$ 使得 $S_{k_j}\xrightarrow{D_i-|q_i|+j} r_i^j$，$1\leqslant j\leqslant |q_i|$，则称 S^* 能满足 q_i 的截止期，记作 $S^*\rightarrow q_i$；反之称 q_i 在 S^* 下错过了截止期。

定义 5(ROBS 问题) 给定实时有序查询集合 $Q=\{q_i\mid 1\leqslant i\leqslant n\}$ 和定义在 DB 上等长广播调度序列集合 A，称在集合 A 中找到一个特定的广播调度 S^{MAX}，使得

$$\forall S^*\in A,\ |S^*\rightarrow q_i|\leqslant |S^{MAX}\rightarrow q_i|,\ 1\leqslant i\leqslant n$$

为实时有序广播调度问题(ROBS: Real-Time Ordered Broadcast Schedule)，其中 $|S^*\rightarrow q_i|$ 是广播调度 S^* 满足 Q 中查询的个数。

定理 1 ROBS 是一个 NP-Hard 问题。

证明：考虑 ROBS 问题的一个特例。每个查询只请求一个数据项($\forall q_i\in Q, |q_i|=1$)且所有查询请求的数据项交集为空($\bigcap_{i=1}^{n} r_i^1=\emptyset$)。在此特例上 ROBS 问题转化为带截止期约束的作业调度问题。文献[14]已证明，对带截止期约束的作业调度问题而言，最小化错过截止期作业数量的判定是 NP-Hard。容易看出，从 ROBS 问题到带截止期约束的作业调度问题转化可在多项式时间内完成，因此 ROBS 问题是 NP-Hard 的。得证。

定理 1 表明给定实时有序查询集合，选择一个广播调度使得错过截止期的查询数量最小这一问题在计算上是难解的。在实际应用中，一方面广播服务器不可能知道未来的查询信息，另一方面也不可能在短时间内完成复杂的计算以进行调度决策，因此本文后续部分重点研究 ROBS 问题的 On-Line 版本及相应的启发式(Heuristic)调度算法。

3 在线(On-Line)的 ROBS 调度算法

3.1 基本思路

在 On-Demand 数据广播环境下，广播服务器需要在线

(On-Line)地进行调度决策。具体说就是每个广播时槽结束前确定下一个广播时槽应发送哪个数据项,调度决策的目标是最小化错过截止期查询的数量。解决这类调度问题的常用策略是根据调度者掌握的信息给每个数据项分派一个优先级,然后在调度时刻选择优先级最高的数据项进行广播。

每个发送到 MSS 上的查询请求 r_i^j 由请求数据项的关键字 K_m,q_i 的截止期 D_i 和 q_i 在数据请求 r_i^j 发出后尚未获得的数据项个数(记为 NoP_i^j)组成。下面用 $\alpha(d_m)$ 表示广播服务器中请求数据项 d_m 的数据请求个数;请求数据项 d_m 的数据请求 r_i^j 的松弛时间(Slack)计算为 D_i-Now,所有请求数据项 d_m 的数据请求中松弛时间的最小值记为 $Slack_m^{\min}$;d_m 对应所有 NoP_i^j 的最小值用 $NoP_m^{\min}$ 表示。

直观分析可以发现下列 3 个因素对查询截止期的满足有影响。

• $\alpha(d_m)$:容易看出发送数据请求个数越大的数据项,越能满足更多的数据访问需求,因此应优先广播 $\alpha(d_m)$ 值大的数据项。

• $Slack_m^{\min}$:在其它因素相同的条件下,发送松弛时间短的请求所对应的数据项,能减小请求该数据项查询错过截止期的可能性。注意,这里使用的是数据项对应所有数据请求松弛时间的最小值,目的是满足时间约束最紧的查询,以免其错过截止期。

• $NoP_m^{\min}$:在其它因素相同的条件下,优先调度剩余数据项少的查询所需的数据项,能使其截止期尽快得到满足。相反地,若这类查询错过截止期,会使已广播的这些查询所需数据项的效用降低。

基于上述分析,提出一种考虑查询语义的 On-Line ROBS 调度算法(OL-ROBS)。对每个数据项 d_m,其调度的优先级由式(1)计算得到:

$$W_m=\alpha(d_m)/Slack_m^{\min}\times NoP_m^{\min} \quad (1)$$

每个数据请求到达时,系统更新每个数据项的 $\alpha(d_m)$,$Slack_m^{\min}$ 和 $NoP_m^{\min}$ 值;在每个调度时槽,服务器选择权值最大的数据项进行发送。

3.2 算法实现

OL-ROBS 算法需在每个广播时槽扫描所有数据项并选择具有最大优先级的数据项广播出去,这使得调度算法的时间复杂度为 O(N)。当数据库中的数据项个数很大或广播带宽很高时(即每个广播时槽的绝对时间相应变短),调度算法的高复杂度会成为系统的瓶颈。例如,若在当前广播时槽内没有执行完调度决策,则无法将最适合的数据项发送出去而影响相应查询的执行。为此,我们根据文献[2]的思想设计了图 2 所示的请求队列数据结构及基于该数据结构的裁减算法,利用裁减算法能显著减少调度算法执行的时间复杂度。

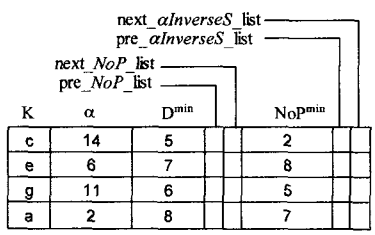

图 2 数据请求管理队列及索引结构

在图 2 中,为表示方便,略去了数据项下标 m。所有数据请求以数据项为单位组成一个队列,该队列中每个队列项由数据项关键字(K)、请求该数据项的所有查询剩余数据请求个数的最小值($NoP^{\min}$)、请求该数据项的数据请求个数(α)和请求该数据项的所有查询截止期的最小值($D^{\min}$)组成。为减少调度时查找队列项的数量,每个队列项还有 4 个指针域 next_NoP_list,pre_NoP_list,next_$\alpha InverseS$_list,pre_$\alpha InverseS$_list。前两个指针域将所有队列项链接成按照 $NoP^{\min}$ 大小降序排列的双向链表 NoP_list;后两个指针域将所有队列项链接成按照 $\alpha/Slack^{\min}$ 降序排列的双向链表 $\alpha InverseS$_list,其中 $Slack^{\min}=D^{\min}-\text{now}$。当有新的数据请求到达时,系统对该队列及链表进行相应的更新。

图 3 给出了如何利用两个有序索引裁减搜索空间。裁减算法首先处理 NoP_list 中的第一个元组(在图 3 中为 d),并根据该元组的 $NoP^{\min}$ 和 $\alpha/Slack^{\min}$ 的值计算出数据项的优先级 $W_d=20\times7=140$,记为 $W_{\max}$。因 NoP_list 和 $\alpha InverseS$_list 都是按照降序排列,所以可以得出在 $\alpha InverseS$_list 链表上 $NoP^{\min}$ 值小于 $W_{\max}/15=9.3$ 的项可以被裁减掉,其中 15 是数据项 d 后下一个数据项 a 的 $NoP^{\min}$ 值。接着裁减算法处理 $\alpha InverseS$_list 上的第一个元组,计算出 $W_f=20\times10=200$,由于 $W_f>W_d$,因此 $W_{\max}$ 更新为 200。同上原因,可以得出在 NoP_list 链表上 $\alpha/Slack^{\min}$ 值小于 $W_{\max}/10=20$ 的项可以被裁减掉,其中 10 为数据项 f 下一个数据项 a 的 $\alpha/Slack^{\min}$ 值。如此反复地在两个链表上执行该裁减算法,直到某个链表上待处理的项超出了计算得到的界限值为止。在图 3 的例子中,处理完队列项 f 即可确定其优先级最高,因此不必再继续处理其他的数据项。从图 3 的例子可以看出,裁减算法可以大大减少队列项的搜索空间。本文第 5 节将通过实验给出裁减算法有效性的证明。

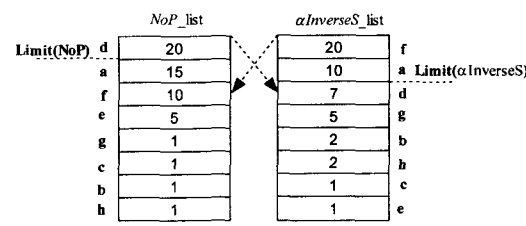

图 3 裁减算法示例

4 模拟实验

4.1 基本思路

为评估 OL-ROBS 及裁减算法的性能,我们设计并使用 C++语言实现了一个仿真实验系统,对比了 OL-ROBS 和目前最为有效的实时数据广播调度算法 Sin-θ[4]。实验的主要性能指标是查询错过截止期的比率。服务器端的数据库大小为 DBSize。客户机的数量为 NumClient。客户机同一时刻只发起一个查询,且只当上一个查询执行完后才继续发起新的查询。每个查询访问 Q 个数据项。为观察查询访问数据项个数对系统性能的影响,设置了查询访问个数区间[Q_{Min},Q_{Max}],每个查询在创建时随机地从该区间中选取其访问数据项个数。查询截止期的设置以数据请求为基本单位,即设置单个数据请求的截止期为 D(在区间 D_{Min} 和 D_{Max} 中随机选择),查询的截止期由公式 $Q\times D+\text{now}$ 计算得到。查询数据访问的模式服从 Zipf 分布。实验的基本时间单位是时槽,每

个时槽广播一个数据项，总的实验时长为 Slots。汇总的实验参数及缺省值如表 1 所列。

表 1　实验参数

变量名	说明	缺省值	取值范围
DBSize	数据库中数据项的大小	2000	1000～16000
NumClient	客户机个数	100	50～400
Q_{Max}	查询访问数据项的最大值	10	5～15
Q_{Min}	查询访问数据项的最小值	2	1～12
D_{Max}	数据请求截止期最大值	80	45～85
D_{Min}	数据请求截止期最小值	40	40～80
θ	Zipf 分布参数	0.8	0.2～1.0
Slots	实验时长	50000	—

4.2　实验结果

在下面的 4 组实验中，分别改变不同的参数来评估 OL-ROBS 的性能。

实验 1(数据库大小和客户机数量对查询错过截止期的影响)

在实验 1 中改变数据库的大小和客户端的数量来观察 OL-ROBS 和 Sin-θ 的性能表现。在图 4 中，SIN100 代表客户机数量为 100 的 Sin-θ 算法，其余依此类推。从图中可以看出，在不同客户机数量下查询错过截止期的比率都随数据库大小的增加而增加，这是因为在有限的下行广播带宽下，数据库大小的增长会延长某个特定数据项两次广播的间隔，从而增加了查询错过截止期的可能性。OL-ROBS 在相同参数设置下的错过截止期比率比 Sin-θ 有显著的降低，下降的百分比最高可达近 30%。性能提高的原因在于 OL-ROBS 综合考虑了查询的数据请求个数、查询截止期和查询剩余数据请求个数等语义信息，而不像 Sin-θ 只考虑单个数据请求的个数和单个数据请求的截止期。

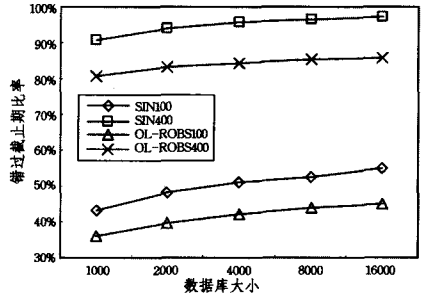

图 4　错过截止期比率 vs. 数据库大小

在图 5 中，SIN1000 代表数据库大小为 1000 的 Sin-θ 算法，其余依此类推。当客户端数为 50 时，两个算法的错过截止期比率均接近 0。当客户机数目不断增加时，两个算法的错过截止期比率均有所增加，但 Sin-θ 增幅大于 OL-ROBS。错过截止期比率随客户机数量的增加而增大是因为随着查询数量的增加，总的数据访问范围也随之增加，有限的广播带宽无法满足所有查询请求，因而造成了越来越多的查询错过截止期。从图 4 还可以发现，随着客户机数量不断增大，错过截止期比率增长的速度逐渐下降，这是因为大多数增加的查询能够在数据广播中得到满足，而这也正是数据广播的多播(Multicast)特性优势所在。同时，该结果和文献[2]中的观察也是一致的。

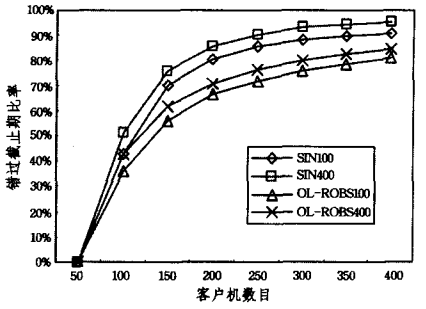

图 5　错过截止期比率 vs. 客户机数目

实验 2(查询访问数据项个数和截止期大小对查询错过截止期的影响)

在实验 2 中改变查询访问数据项个数和查询截止期大小来观察 OL-ROBS 和 Sin-θ 的性能表现。在图 5 中，SIN100 代表客户机数量为 100 的 Sin-θ 算法，其余依此类推。在图 6 中，两个算法在不同参数设置下的错过截止期比率都随数据项截止期区间变大而降低，这是因为截止期的增大使得查询有更多的时间来等待所需的数据。与实验 1 结果相同，OL-ROBS 在绝大多数参数设置下有比 Sin-θ 更好的性能。一个例外是当数据项截止期在 80～85 时，OL-ROBS 表现比 Sin-θ 略差。但由于该设置下两者的错过率都非常低且差别很小，因此整体上看仍是 OL-ROBS 表现得最好。

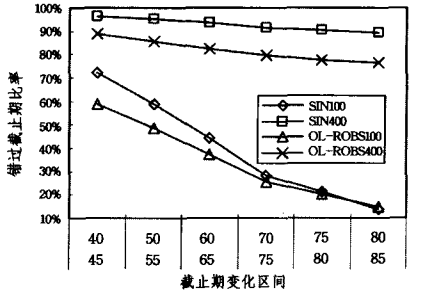

图 6　错过截止期比率 vs. 截止期区间变化

在图 7 中，两个算法的错过截止期比率随查询访问数据项个数的增大而提高，这是因为在相同的截止期范围设置下，访问数据量的增大提高了查询错过截止期的概率。当客户机数量和查询请求数据项个数过大时(分别为 400 和 12)，两个算法错过截止期比率较接近且都趋于 1。由于实际数据广播应用中查询访问对象的数量不会太大[15]，因此 OL-ROBS 有比 Sin-θ 更好的可用性。

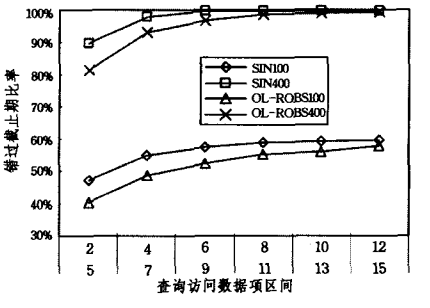

图 7　错过截止期比率 vs. 查询访问数据项数目

实验 3(Zipf 参数对查询错过截止期比率的影响)

实验 3 通过改变查询的数据访问模式来评估 OL-ROBS 和 Sin-θ 的表现，其中 SIN100 代表客户机数量为 100 的 Sin-θ 算法，其余依此类推。从图 8 可以看出，随着 Zipf 参数不断

趋近于1，两种算法错过截止期的比率都有下降，这是因为访问模式的集中有利于“流行”的数据项获得更高的调度优先权，从而降低访问这些数据的查询错过截止期的可能性。与我们预期的相同，OL-ROBS具有更好的性能表现。

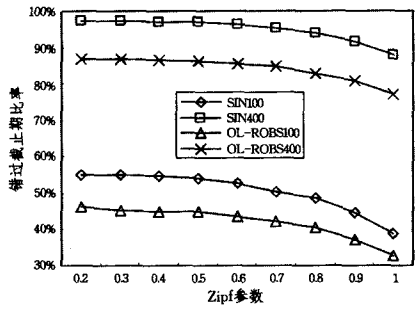

图8 错过截止期比率 vs. Zipf 参数

实验4(裁减算法缩减搜索空间的有效性)

该实验中比较了使用裁减算法和不使用裁减算法时，为找到最大的 W 值所搜索数据项的个数。实验结果如图9所示，其中纵轴为使用裁减算法和不使用裁减算法时搜索队列项的比值。在实验中变化了数据库的大小，并在不同客户机数量的设置下进行了实验。从结果可以看出，裁减算法极大地减少了请求队列搜索的范围，最坏情况下也减少了近80%的搜索空间。随着数据库大小的增加，裁减算法的性能变得更为有效，这可以解释为数据库内数据项的增加使得“热点”数据变得相对更少，因此裁减算法可以更快速地缩小搜索界限。同样地，在数据库大小相同的设置下，客户机越多，“热点”数据变得也相对越少，同样的原因导致搜索数据项比率下降。

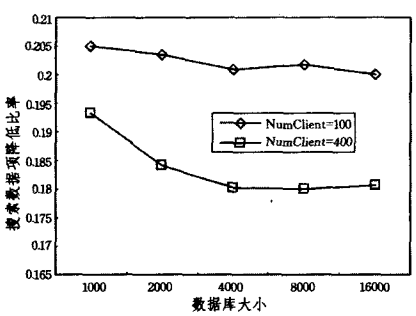

图9 裁减比率 vs. 数据库大小

5 相关工作

文献[11]基于对移动通信网络非对称性(Asymmetric)的分析，提出了单个广播频段上基于数据访问频率的数据广播组织方法，并首次使用了广播磁盘(Broadcast Disks)概念。文献[12]提出了缓存失效和预取两种策略，并通过实验证明将这两种策略与文献[1,5]中广播组织方法相结合，能够有效地解决更新分发的问题。文献[2]的研究表明，在用户数量和访问模式经常变化的应用场合，On-Demand 数据广播具有更高的带宽利用率和可扩展性。文献[13]对 Push-Based 和 On-Demand 数据广播相结合的混合数据广播进行了研究，但其采用投硬币方式决定数据项以哪种方式来发送却过于简单，且对硬币正反参数如何确定也没有提及。On-Demand 数据广播的核心问题是如何调度数据请求，文献[2]提出了一种基于数据请求个数和数据请求等待时间的调度算法，通过模拟实验表明其在大规模数据分发时具有更低的平均响应时间。

上述数据广播研究中，主要性能指标是平均等待时间和调谐时间。在许多应用中，往往不只要求平均等待时间短，还希望数据项在给定的时间段内就能发送出来，即移动应用具有显式的时间约束，称这一类数据广播为实时数据广播。实时数据广播中，虽然平均等待时间和调谐时间仍然重要，但移动应用错过截止期的比率和数据项的时效性成为首要指标。文献[3]研究了 On-Demand 广播环境下有截止期约束的数据请求的调度算法，通过对比 Round-Robin，FCFS，EDF 等几种经典调度方法，发现 EDF 表现最好。文献[4]在文献[3]的基础上，综合考虑数据请求的截止期和数据请求的个数，提出了 Sin-θ 调度算法，实验证明了该算法的优越性。文献[7]研究了当数据项不是等大小情况下的广播调度算法。在上述几种算法中，Sin-θ 由于既考虑了数据请求个数又考虑了数据请求的松弛时间，因此有最好的性能。

不考虑时间约束的数据广播环境下查询处理问题也已经得到许多研究，文献[6,15]研究了针对有序(Ordered)查询处理的数据广播组织和调度问题。文献[15]关注的磁盘组织基本对象是单个数据项。与此不同，文献[9]将关系数据库中的关系或面向对象数据库中的对象作为磁盘组织的基本单位，使用图来描述查询模式，并基于分枝界限法设计了搜索算法，以寻找最优的广播磁盘组织。但这些研究都基于 Push-Based 广播模式且没有考虑实时约束。

从已有相关工作可以看出，虽然非实时数据广播环境下的查询处理和单数据项的实时数据广播技术已分别得到许多研究，但同时考虑查询语义及实时约束的数据广播技术尚未得到学术界深入研究。就我们所知，本文第一次提出了数据广播环境下考虑实时约束的查询处理问题，并给出了有效的解决方案。

结束语 本文研究了 On-Demand 数据广播环境下实时有序查询处理问题，本文工作的主要贡献包括：

1)基于对实时有序查询处理问题的分析，定义了一类新的调度问题 ROBS 并证明了该问题的 Off-Line 版本是 NP-Hard 的。

2)提出了一种新的考虑查询语义的 On-Line 调度算法 OL-ROBS，算法综合考虑了数据请求个数、查询截止期和查询剩余数据请求个数等语义信息。

3)设计了一种提高调度算法执行效率的裁减算法，该算法可有效地减少调度决策时的搜索空间。

4)通过模拟实验将 OL-ROBS 与目前最为有效的实时数据请求调度算法 Sin-θ 进行了比较，结果显示 OL-ROBS 具有更好的性能。

参 考 文 献

[1] Acharya S, Muthukrishnan S. Scheduling on - demand broadcasts: new metrics and algorithms[C]// Proceedings of ACM MobiCom'98. 1998

[2] Aksoy D, Franklin M. RxW: a scheduling approach for large scale on-demand data broadcast[J]. IEEE/ACM Trans. Networking, 1999, 7(3): 846-860

[3] Xuan P, Subhabrata S, Oscar G, et al. Broadcast on demand: efficient and timely dissemination of data in mobile environments [C]// Proceedings of RTAS. 1997

(下转第186页)

因此，Einstein 谜可以表达为：

$$(1)\wedge(2)\wedge\cdots\wedge(17)\wedge(18) \tag{19}$$

求谁养鱼即求满足式(19)的解。由于满足式(19)的解不一定唯一，因此可能存在着多种可能性的解。

3 基于 MinSat 的求解

使用 MinSat 求解器需先将要求解的公式等价地转换成合取范式(CNF)，然后以 DIMACS 的格式表达。为了将公式表达成 DIMACS 格式，先对公式中所涉及的布尔变量定义索引，如表 2 所列。

表 2 布尔变量索引的定义

房间	颜色	索引	国籍	索引	宠物	索引	饮料	索引	香烟	索引
	a111	1	a121	26	a131	51	a141	76	a151	101
	a112	2	a122	27	a132	52	a142	77	a152	102
#1	a113	3	a123	28	a133	53	a143	78	a153	103
	a114	4	a124	29	a134	54	a144	79	a154	104
	a115	5	a125	30	a135	55	a145	80	a155	105
	a211	6	a221	31	a231	56	a241	81	a251	106
	a212	7	a222	32	a232	57	a242	82	a252	107
#2	a213	8	a223	33	a233	58	a243	83	a253	108
	a214	9	a224	34	a234	59	a244	84	a254	109
	a215	10	a225	35	a235	60	a245	85	a255	110
	a311	11	a321	36	a331	61	a341	86	a351	111
	a312	12	a322	37	a332	62	a342	87	a352	112
#3	a313	13	a323	38	a333	63	a343	88	a353	113
	a314	14	a324	39	a334	64	a344	89	a354	114
	a315	15	a325	40	a335	65	a345	90	a355	115
	a411	16	a421	41	a431	66	a441	91	a451	116
	a412	17	a422	42	a432	67	a442	92	a452	117
#4	a413	18	a423	43	a433	68	a443	93	a453	118
	a414	19	a424	44	a434	69	a444	94	a454	119
	a415	20	a425	45	a435	70	a445	95	a455	120
	a511	21	a521	46	a531	71	a541	96	a551	121
	a512	22	a522	47	a532	72	a542	97	a552	122
#5	a513	23	a523	48	a533	73	a543	98	a553	123
	a514	24	a524	49	a534	74	a544	99	a554	124
	a515	25	a525	50	a535	75	a545	100	a555	125

根据索引的定义，一个 CNF 公式很容易写成 DIMACS 格式。经过 MiniSat 的执行得到唯一的结果：

$a_{113}=1, a_{125}=1, a_{131}=1, a_{143}=1, a_{152}=1,$

$a_{215}=1, a_{223}=1, a_{234}=1, a_{244}=1, a_{253}=1,$

$a_{311}=1, a_{321}=1, a_{333}=1, a_{342}=1, a_{351}=1,$

$a_{414}=1, a_{422}=1, a_{435}=1, a_{445}=1, a_{455}=1,$

$a_{512}=1, a_{524}=1, a_{532}=1, a_{541}=1, a_{554}=1$

根据表 1，得知 Einstein 谜的解为德国人养鱼，如表 3 所列。

表 3 Einstein 谜的结果

	房间#1	房间#2	房间#3	房间#4	房间#5
颜色	黄	蓝	红	绿	白
国籍	挪威	丹麦	英国		瑞典
宠物	猫	马	鸟		狗
饮料	水	茶	牛奶	咖啡	啤酒
香烟	Dunhill	Blend	Pall Mall	Prince	Blue master

结束语 通过将 Einstein 谜转换为 SAT 求解问题，可以将复杂的逻辑推理题转换成命题逻辑的 SAT 问题，并通过 SAT 求解器自动解答问题。这一结论体现了现实生活中的一些复杂问题可以通过成熟的逻辑、数学理论和方法严格求解。

参 考 文 献

[1] Huth M, Ryan M. Logic in Computer Science , Modelling and Reasoning About Systems(Second Edition)[M]. Cambridge University Press, ISBN 0 521 54310X

[2] Yeomans J. Solving "Einstein's Riddle" Using Spreadsheet Optimization[J]. Informs Transaction on Education, 3(2): 55-63

[3] MiniSat SAT Solver[EB/OL]. http://minisat.se/

[4] Gu J, Purdom P W, Franco J, et al. Algorithms for the Satisfiability (SAT) Problem: A Survey[M]. Satisfiability Problem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1997: 19-152

[5] Cook S A. The complexity of theorem-proving procedures[C]// Proceedings of the third ACM Symposium on Theory of Computing. 1971: 151-158

[6] Crawford J M, Auton L D. Experimental results on the crossover point in satisfiability problems[C]// Proc. of AAAI93. Aug 1993: 21-27

(上接第 150 页)

[4] Xu Jianliang, Tang Xueyan, Lee Wang-Chien. Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation[J]. IEEE Trans. Parallel Distrib. Syst., 2006, 17(1): 3-14

[5] Chung Y D, Kim M-H. QEM: a scheduling method for wireless broadcast data[C]// Proceedings of DASFAA. 1999

[6] Chehadeh Y C, Hurson A R, Kavehrad M. Object organization on a single broadcast channel in the mobile computing environment[J]. Multimedia Tools and Applications, 1999, 9(1): 69-94

[7] Lee Victor C S, Wu Xiao Joseph, Ng Kee-Yin. Sheduling real-time requests in on-demand data broadcast environments[J]. Real-Time Systems, 2006, 34(2): 83-99

[8] Wang Hongya, Ning Guoqin, Li Guohui, et al. Mobile Real-time Read-only Transaction Processing in Broadcast Disks[J]. Journal of Information Science and Engineering, 2006, 22(5): 1249-1264

[9] Si A, Leong H V. Query optimization for broadcast database [J]. Data and Knowledge Eng., 1999, 29(3): 351-380

[10] Chung Yu-Chi, Chen Chao-Chun, Lee Chiang. Design and performance evaluation of broadcast algorithms for time-constrained data retrieval[J]. IEEE Trans. Knowl. Data Eng., 2006, 18(11): 1526-1543

[11] Acharya S, Franklin M, Zdonik S. Broadcast disks: data management for asymmetric communication environments[C]// Proceedings of ACM SIGMOD. 1995

[12] Acharya S, Franklin M, Zdonik S. Disseminating updates on broadcast disks[C]// Proceedings of VLDB. 1996

[13] Acharya S, Franklin M, Zdonik S. Balancing push and pull for data broadcast[C]// Proceedings of ACM SIGMOD. 1997

[14] Acharya S, Franklin M, Zdonik S. Balancing push and pull for data broadcast[C]// Proceedings of ACM SIGMOD. 1997

[15] Huang Jiun-Long, Chen Ming-Syan, Peng Wen-Chih. Broadcasting Dependent Data for Ordered Queries Without Replication in a Multi-channel Mobile Environment[C]// Proceedings of ICDE. 2003