移动实时环境下一种改进的广播调度算法

帖 军 王小荣 蒋天发
(中南民族大学计算机科学学院 武汉 430074)

摘 要 在网络带宽不对称的移动实时环境中,数据广播是一种有效的数据访问方式。针对这种网络特性,分析了现今已经存在的某些广播调度算法。针对UFO算法,分别提出了SBS算法和CRS算法,它们从服务器、移动客户端两个方面进行了改进。两种算法可以根据给定的数据项访问概率分布,自动生成广播调度。通过理论分析和实验结果表明,该算法不会产生事务重启,并且可以有效减少数据的访问时间,使用户访问数据广播的平均等待时间最小。

关键词 移动实时环境,数据广播,广播调度算法
中图法分类号 TP392 **文献标识码** A

Improved Broadcast Scheduling Algorithm in Mobile Real-time Environment

TIE Jun WANG Xiao-rong JIANG Tian-fa
(College of Computer Science, South-Central University for Nationalities, Wuhan 430074, China)

Abstract Data broadcast is an efficient method for data accessing in the asymmetry bandwidth of mobile real-time environment. For characteristics of such a network, we analyzed some of existing broadcast scheduling algorithms, such as UFO algorithm and propose SBS algorithm and CRS algorithm. They improve UFO from server and mobile client. The two algorithms can automatically generate broadcast scheduling lists which depend on the given data items' probability distribution. Then theoretical analysis and experimental results show that the proposed algorithm can not produce any transactions' restarting and effectively reduce data items' accessing time. All of these make the average waiting time that users access data broadcast minimized.

Keywords Mobile read-time environment, Data broadcast, Broadcast scheduling algorithm

1 引言

移动实时计算技术的发展,使得用户可以在任意地点、任意时间访问信息,它正逐渐成为一种流行的计算方式。与传统固定网络相比较,支持移动实时计算的无线通信网络具有通信网络不稳定、上下行带宽相差大的特点,并且移动计算设备往往只有很小的存储能力,且电源能量有限。为了支持大量移动终端设备并发访问服务器并节省电源的消耗,人们提出的服务器向空中广播数据,移动终端从空中获取数据的数据发送接收模式,即数据广播[1]。图1为通用的数据广播体系结构。

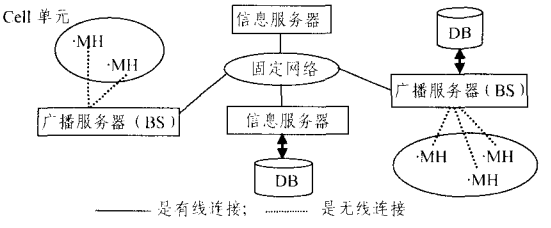

图1 通用的数据广播体系结构

基于数据项在带宽中所占的比例,文献[2]分析了平坦调度策略和非平坦调度策略;基于事务冲突处理,文献[3]提出了UFO(update-first with order)算法;基于实时环境的单数据项请求情况,文献[4]提出了EDF(Earliest Deadline First)算法;文献[5]提出了MRF(Most Request First)算法;文献[6]提出了SIN(Slack Time Inverse Number of Pending Request)算法;基于多数据项请求的情况提出了很多算法,其中面向请求的有文献[7]提出了QEM(Query Distance Expanding Method)算法和文献[8]提出的GCM(Grey Code Clustering Method)算法;面向数据的有文献[9]提出的访问图算法和文献[10]提出的从数据联系角度考虑的调度算法;文献[11]提出了采用遗传算法的调度算法;文献[1]介绍了On-demand模式下的数据项广播调度算法。这些调度算法都有自己的优缺点,其中UFO算法可以保证移动客户端读到的数据都是最新的并且能够保证数据库的一致性,但是未考虑到事务处理的实时性能。当更新事务频繁发生时,会使得客户端的访问时间无限增长。

本文通过分析不同事务在移动实时环境中广播调度时的特性,提出了两个改进的广播调度算法,并通过仿真模型与

到稿日期:2011-06-21 返修日期:2011-09-15 本文受中南民族大学中央高校基本科研业务费专项资金项目(CZY11007)资助。
帖 军(1976—),男,副教授,硕士生导师,主要研究方向为移动计算、分布式系统,E-mail:musa@scuec.edu.cn;王小荣(1988—),女,硕士生,主要研究方向为移动实时数据库;蒋天发(1954—),男,教授,硕士生导师,主要研究方向为高性能网络与信息安全,E-mail:jiangtianfa@163.com(通信作者)。

UFO算法比较，发现其显著缩短了数据的访问时间，并且降低了事务的重启次数，从而提高了移动实时数据库广播调度的效率。

2 相关定义

广播服务器(Broadcast Server，BS)为了方便广播数据，通常将待广播的数据项存放在固定的物理磁盘上，再按照一定的广播调度算法进行统一调度构成逻辑磁盘。逻辑磁盘按照数据项的访问频度划分不同的块，同一块逻辑磁盘上数据的总体访问频率相同。其中一些磁盘块因为存放数据比较少，在单位时间内其数据项被访问频度高而被称为快速磁盘，反之则被称为慢速磁盘。经常被访问的数据项称为热数据项，不常被访问的数据项称为冷数据项。

定义 1(广播次周期) 将数据项按访问频度进行划分，访问频度相近的数据项放在同一个广播磁盘上，如热数据项位于快速磁盘上，冷数据项放在慢速磁盘上。则一个广播次周期 $SubCycle$ 可以定义为 $SubCycle=\sum_{i=1}^{n}T(d_{ij})$。其中 n 为广播磁盘数，d_{ij} 为广播磁盘 i 上的数据项 j，函数 $T(x)$ 为广播数据项 x 所需要的时间。每个广播次周期的长度相等。

定义 2(广播补偿周期) 一个补偿周期 $ReCycle$ 是所有次周期结束后、新一轮次周期到来前进行数据广播的补偿时间。

定义 3(广播周期) 广播周期 $BCycle=\sum_{j=1}^{n}SubCycle_j+ReCycle$，$n$ 为某个存放广播数据项最多的广播磁盘中数据项的个数。即一个广播周期由 n 个广播次周期和一个广播补偿周期组成。

定义 4(移动只读事务) 一个移动只读事务可以表示为一个五元组 $MRT=\langle MT,Data,Op,Evn,F\rangle$，其中

$MT=\{MobileTransaction\mid MobileTrancaction$ 是一个可以在 BS_k 间自由移动的移动事务，$1\leqslant k\leqslant n$，n 为一个移动实时数据库(MRTDB)中 BS 的个数$\}$；

$Data=\{DataItem\mid DataItem$ 是 MT 操作的数据项$\}$；

$Op=\{Operation\mid Operation=read\cup write\cup sent\}$；

$Evn=\{Event\mid Event=READ\cup UPDATE\cup BROADCAST\}$；

对 $\forall x\in Data$，$Op(x)=read\rightarrow Evn(MT)=READ$；

F 为 MT 发生 $READ$ 事件下，$Data$ 与 Op 满足的函数关系：$F:(Evn(MT),Data,Op)\rightarrow read$。

定义 5(更新事务) 一个更新事务可以表示为一个五元组 $UT=\langle T,Data,Op,Evn,F\rangle$，其中

$T=\{Transaction\mid Transcation$ 是一个事务$\}$；

$Data=\{DataItem\mid DataItem$ 是 T 操作的数据项$\}$；

$Op=\{Operation\mid Operation=read\cup write\cup sent\}$；

$Evn=\{Event\mid Event=READ\cup UPDATE\cup BROADCAST\}$；

对于 $\forall x\in Data$，$Op(x)=write\rightarrow Evn(MT)=UPDATE$；

F 为 T 发生 $UPDATE$ 事件下，$Data$ 与 Op 满足的函数关系：$F:(Evn(T),Data,Op)\rightarrow write$。

UT 可以发生在服务器中，也可以发生在 MH 中。若在服务器中，则不能移动；在 MH 中可以自由移动，当移动时，需要考虑过区切换的问题，处理比较复杂。本文只讨论 UT 在服务器端发生的情况。

定义 6(广播事务) 一个广播事务可以表示为一个五元组 $BT=\langle T,Data,Op,Evn,F\rangle$，其中

$T=\{Transaction\mid Transaction$ 是一个事务$\}$；

$Data=\{DataItem\mid DataItem$ 是 T 操作的数据项$\}$；

$Op=\{Operation\mid Operation=read\cup write\cup sent\}$；

$Evn=\{Event\mid Event=READ\cup UPDATE\cup BROADCAST\}$；

对于 $\forall x\in Data$，$Op(x)=sent\rightarrow Evn(MT)=BROADCAST$；

F 为 T 发生 $BROADCAST$ 事件下 $Data$ 与 Op 满足的函数关系：$F:(Evn(T),Data,Op)\rightarrow sent$。

BT 只允许在某个 BS 上发生并完成，没有移动性。

定义 7(读集合) 一个事务的读集合可以表示为一个三元组 $RD=\langle Data,Op,F\rangle$，其中

$Data=\{DataItem\mid DataItem$ 是 T 可操作的所有数据项$\}$；

$Op=\{Operation\mid Operation=read\cup write\}$；

F 为 $Data$ 与 Op 满足的函数关系：$F:(Data,Op)\rightarrow read$。

定义 8(写集合) 一个事务的写集合可以表示为一个三元组 $WD=\langle Data,Op,F\rangle$，其中

$Data=\{DataItem\mid DataItem$ 是 T 可操作的所有数据项$\}$；

$Op=\{Operation\mid Operation=read\cup write\}$；

F 为 $Data$ 与 Op 满足的函数关系：$F:(Data,Op)\rightarrow write$。

定义 9(补偿广播事务) 在一个补偿周期 $ReCycle$ 内，补偿广播事务 $ReBT$ 可以表示为一个四元组 $ReBT=\langle BT,Data,Op,F\rangle$，其中

$BT=\{BroadcastTransaction\mid BroadcastTransaction$ 是一个广播事务$\}$；

$Data=\{DataItem\mid DataItem$ 是 BT 操作的数据项，$DataItem\in RD(BT)\cap(\bigcup_{i=1}^{n}WD(UT_i))\}$，其中 BT 的执行时间为本次广播中所有广播次周期的执行时间之和，n 为在 BT 执行期间发生的 UT 的个数；

$Op=\{Operation\mid Operation=read\cup write\cup sent\}$；

F 为 BT 事件下 $Data$ 与 Op 满足的函数关系：$F:(Data,Op)\rightarrow sent$。

$ReBT$ 只允许在某个 BS 上发生，并且仅仅紧随 BT 完成后发生。

3 广播调度算法

本文分别对数据广播的服务器端和移动客户端进行分析，提出了两种算法，即服务器广播调度算法和客户端接收广播调度算法。

3.1 服务器广播调度(Server Broadcast Scheduling，SBS)算法

在移动实时环境中，MH 与 BS 之间是通过无线网络进行通信的。由于上下行带宽的巨大差异，使得数据广播成为解决这个问题的有效方法。在 UFO 算法中提出了在服务器端发生 BT 与 UT 冲突时，直接重启 BT，在 UT 执行完后，再

执行 BT。这样虽然可以保持 BT 广播到的数据都是最新的，但是当大量 UT 发生时，BT 就会长时间不能执行，从而 MT 不得不等待很长时间，不符合移动实时数据库的要求。

本文提出的 SBS 算法中，BS 将每个广播磁盘 i 上的数据项存放在一个循环队列 Q_i 中，其长度等于该磁盘上数据的个数，$front$ 和 $rear$ 分别为队头和队尾，BS 轮流取每个循环队列 Q_i 队头指向的值 d_{ij}，并将其广播发送给 $BScell$ 中的每个 MH。再构造一个空队列 U。若在广播过程中有 UT 发生，那么此时 BT 与 UT 并行执行。BT 继续广播自己的数据项，直到广播结束，而 UT 则使用两级复制的方法执行自己的操作，一个 UT 执行完毕后同步更新 WD 和 DB，并往队列 U 中添加更新节点，节点信息包含新值和位置相关信息。当 BT 广播完毕后，紧接着执行一个 $ReBT$，广播队列 U 中每个数据项，直至 U 被访问完。这样一个周期的数据广播在服务器端就完成了。

SBS 算法的实现如下所示。在每个广播周期中都会执行如下过程。其中代码中的 i 都表示广播磁盘的个数；j 都表示广播磁盘 i 上的数据项 j，共有 n 个广播磁盘。

```
Initialize_BTList()
{ for i from 1 to n
   { InitQueue(Queue &Qi);
     while data in diski havn't been accessed
     {EnQueue(Queue &Qi,ElemType dij);}}
}//end Initialize_BTList
Initialize_ReBTList()
{ InitQueue(Queue &U);
}//end Initialize_ReBTList
BT()
{set m:= Qi. MaxSize;
  for j from 1 to m
  { for i from 1 to n
    { set t:= Qi. front;
      ElemType dit:=PeekQueue(Queue Qi);
      Qi. front move next position;
      Broadcast(dit);}}
}//end BT
UT()
{ UT use Two-Tier Replication Method;
  update WD and DB in the same time;
  for k from 1 to WD. MaxSize
    if dij updated
      Add a node in U for dij;
}//end UT
ReBT()
{ for j from 1 to U
  { ElemType dij= PeekQueue(Queue U);
    Broadcast(dij);}}
}//end ReBT
SBSscheduler()
{ Initialize_BTList();
  Initialize_ReBTList();
    BT()∩UTx();//∩ stand for concurrent
  ReBT()∩UTy();//UTx and UTy are different UTs
}//end scheduler
```

3.2 客户端接收广播调度(Client Receive Scheduling,CRS)算法

在 UFO 算法中，MH 一直监听广播。当 BS 将广播发送到 $cell$ 中，MH 对比此时正广播的数据是否需要。若不需要，则一直等待，直到监听到需要的数据被广播。当服务器端 BT 被重启，在 MH 端，BT 立即重启，MT 也需要重启，直到 BT 重新广播。这样使得事务的重启率过高，严重浪费资源。

本文提出的 CRS 算法中，MH 端构造一个队列 R 用于存放数据请求，再构造 n 个循环队列 S_i 用于存放接收到的广播数据项 d_{ij}，S_i 的长度与服务器端相应的 Q_i 的长度相等。将监听到的数据项 d_{ij} 存放在 S_i 的 j 位置上。若该位置已经存有数据，则覆盖旧值；若无，则直接填充。当一个周期的广播监听完后，则遍历队列 R，通过 R 中结点存放的请求数据项的位置信息在 S 的相应位置取值并送往 MH，直到 MH 的所有数据请求都满足为止。

CRS 算法的实现过程如下所示。在每个广播周期中都会执行该过程。其中代码中的 i 都表示广播磁盘的个数；j 都表示广播磁盘 i 上的数据项。

```
CRSscheduler()
{Initialize_RequireList()∩Receive_Data();
  MT();
}//end CRSscheduler
Initialize_RequireList()
{ InitQueue(Queue &R);
  while exists require(dij)
    Add a node in R for dij;}
}//end Initialize_RequireList
Receive_Data()
{ for i from 1 to n
  { InitQueue(Queue &Si);
    Si. MaxSize:= Qi. MaxSize;}
  for k from 1 to RD. MaxSize
    if dij exist in Si &&dij has saved in j'
      Si[j']:= dij;
}//end Receive_Data()
MT()
{ for i from 1 to R. MaxSize
    peek sij based on R[i]
    read(sij);
}//end MT()
```

4 仿真实验

4.1 实验模型及实现

为了评估 SBS 算法和 CRS 算法的有效性，本文设计并实现了仿真实验系统。实验的主要性能指标是 MT 的平均访问时间和 MT 的夭折率。采用定量分析的实验方法，对在 MH 上使用基于 SBS 算法、CRS 算法和 UFO 算法的性能进行比较。然后将数据项分为热数据项和冷数据项后，再测试 CRS 算法本身性能，并分析其优、缺点。

模拟的模型由服务器、移动主机和广播磁盘组成。广播采用聚类磁盘组织。移动只读事务一直执行到提交，即使错过截止时间。实验中用到的主要参数包括 MT 个数、冷热数据项访问率、修改率、RD 和 WD 中数据项的个数、UT 服从泊松分布及 BT、UT、MT 之间优先级的设置和对广播磁盘相关项的设置等。实验参数及其取值如表 1 所列。

表 1　仿真实验参数

(a) 服务器端参数设置

参数	含义	数值
RD	RD 数据项个数	500
WD	WD 数据项个数	500
UT Distribute	UT 的概率分布	X~P(1)
Seize HotData	RD 中热数据项比例	0.1
Update HotData	热数据项平均修改率	0.7
Priority(BT,UT)	BT 与 UT 优先级设置	UT is higher

(b) 广播参数设置

参数	含义	数值
No. Broadcast Dataitems	广播数据项总个数	500
No. Channels	信道数	1
Dataitems TimeInterval	广播数据项时间间隔	0.01s
No. Broadcast Disks	广播磁盘数	2
BTi TimeInterval	BT 间时间间隔	1s

(c) MH 端参数设置

参数	含义	数值
No. MT	MT 的个数	5
Think Time	事务到达时间间隔	1s
Require TimeInterval	数据请求时间间隔	0.1s
Deadline	截止期	4s
Priority(BT,MT)	BT 与 MT 优先级置	BT is higher

4.2　实验结果

通过仿真模拟评测本文提出的 SBS 算法和 CRS 算法性能。将 CRS 调度算法与 UFO 调度算法进行比较，来评价本文提出的 SBS 算法和 CRS 算法，然后将 *MT* 中的数据项分为冷数据项和热数据项，再评价 SBS 算法和 CRS 算法自身。

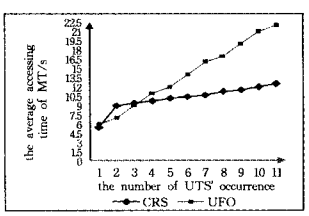

图 2　UT 下 MT 的平均访问时间

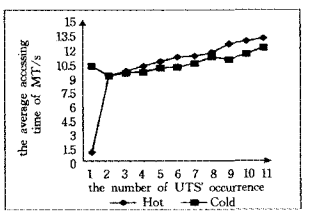

图 3　UT 下 CRS 算法中 MT 冷热数据平均访问时间

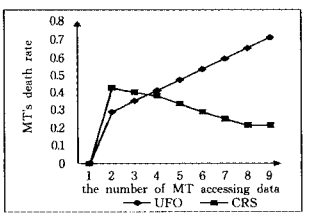

图 4　MT 访问量下的 MT 夭折率

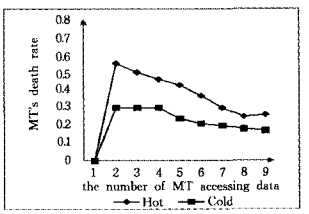

图 5　CRS 算法中 MT 冷热数据访问量下的 MT 夭折率

在性能评价模型中，*RD* 中数据项的个数与 *WD* 中数据项的个数和值都相等。本文提出的算法在广播服务器端采用补偿广播事务的做法，所以不会出现 *BT* 的重启和 *MT* 的重启，而 *UFO* 算法则会因为 *UT* 的原因使得 *BT* 和 *MT* 不断重启。图 2 表示在一个 *MT* 中服务器端发生不同个数 *UT* 时，*MT* 在最坏情况下的平均访问时间。通过比较可以看出，UFO 算法在 *UT* 发生个数较多的情况下，*MT* 的平均访问时间急剧上升。而在 CRS 算法下，*MT* 的平均访问时间虽然也是随着 *UT* 发生个数的增加而增加，但是增加的幅度远小于 UFO 算法。图 3 表示在 CRS 算法下将数据项分为冷数据和热数据时的平均访问时间的比较。可以看出，当 *UT* 发生较少时，热数据项所需要的访问时间远小于冷数据项。但是当 *UT* 单位时间内发生过多时，冷数据项因为自身被访问更改次数少而使得其访问时间增长率低于热数据项。图 4、图 5 表示在一次广播周期中发生 10 次 *UT* 时，随着 *MT* 访问数据量的增加 *MT* 的夭折率变化情况。由图 4 可知，UFO 算法随着数据量的增加 *MT* 的夭折率急剧上升，而本文提出的算法却随着数据量的增加 *MT* 的夭折率降低。图 5 表示，随着访问数据项中冷、热数据项的增加，CRS 算法中冷数据项的夭折率远低于热数据项的夭折率。

结束语　本文提出的广播调度算法，分别对服务器和移动客户端的算法进行了改进，使得 *MH* 端不需要因为 *BT* 的重启而等待。通过仿真模拟可以看出，随着 *UT* 发生个数和 *MT* 数据访问量的增加，本文算法性能显著好于其他广播机制。

参 考 文 献

[1]　张卓瑶，孙未未，余平，等. 无线环境中多数据项广播调度算法综述[J]. 计算机科学，2009(5)：16-20

[2]　Litman A, Moran-Schein A. On Distributed Smooth Scheduling [C]// Proceeding of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 05). 2005:76-85

[3]　Lam K Y, Chan E, Au Mei-wai. Broadcast of Consistent Data to Read-only Transactions from Mobile Clients[C]// Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications. New Orleans, 1999, 02

[4]　Haritsa J R, Livny M, Carey M J. Earliest deadline scheduling for real-time database systems[C]// Proceedings of the 12th IEEE Real-Time Systems Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1991: 232-243

[5]　Wong J W. Broadcast delivery [J]. Proceedings of the IEEE, 1998, 76(12): 1556-1557

[6]　Xu Jian-liang, Tang Xue-yan, Lee W C. Time-critical On-demand Data Broadcast: Algorithms, Analysis, and Performance Evaluation[J]. IEEE Trans. on Parallel Distrib. Syst, 2006, 17(1): 3-14

[7]　Chung Y D, Kim M H. QEM: A Scheduling Method for Wireless Broadcast[C]// International Conference on Database Systems for Advanced Applications. 1999

[8]　Lee J Y, Chung Y D, Lee Y J, et al. Grey Code Clustering of Wireless Data for Partial Match Queries[J]. Journal of Systems Architecture, 2001(47): 445-458

[9]　Lee G, Lo S C. Broadcast Data Allocation for Efficient Access of Multiple Data Items in Mobile Environments[J]. Mobile Networks and Applications, 2003(8): 365-375

[10]　Chung Y D, Bang S H, Kim M H. An Efficient Broadcast Data Clustering Method for Multipoint Queries in Wireless Information Systems[J]. The Journal of Systems and Software, 2002 (64): 173-181

[11]　Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning[M]. ADDison Weskey, 1989