移动实时数据库 QoS 管理和更新事务调度算法*)

向 军[1,2] 李国徽[1] 杨 兵[1] 杜建强[3]
(华中科技大学计算机学院 武汉 430074)[1] (湖北民族学院信息工程学院 恩施 445000)[2]
(江西中医学院计算机系 南昌 330006)[3]

摘 要 分布式环境下的移动实时数据库服务应用逐渐广泛,但系统负载的不可预测甚至可能的超载使得应用受到限制。系统中事务在低带宽移动环境下竞争有限的系统资源导致重启或夭折,从而给系统带来损失甚至灾难。通过结合时间有效性和值域的有效性提出基于的一种新的衡量系统性能的参数标准,如错过截止期事务的比率、数据新鲜度、CPU 利用率去保证 QoS,同时提出一种更新事务调度新算法和基于反馈控制的架构去实现。通过仿真实验评估,算法可以从稳定性能和暂态性能保证系统的性能指标不会超过由数据库管理员事先提出的 QoS 规范。

关键词 导出数据,服务质量,移动实时数据库,更新事务

Scheduling Algorithm of Update Transactions and Quality of Service Management in Real-time and Mobile Database Systems

XIANG Jun[1,2] LI Guo-hui[1] YANG Bing[1] DU Jian-qiang[3]
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China) [1]
(School of Information Engineering, Hubei Institute for Nationalities, Enshi 445000, China) [2]
(Department of Computer sicence, Jiangxi University of Traditional Chinese Medicine, Nanchang 330006, China)[3]

Abstract The demand on mobile and real-time database application and service in distributed environment is becoming increasingly extensive, since the workload cannot be precisely predicted so that they can become overloaded possibly. Low bandwidth in mobile environment will lead to aborting or restarting for transactions because of competing for the limited system resources severely, resulting in damage or a catastrophe possibly. In this paper, we propose a novel performance metrics for example deadline miss ratio of transactions and data freshness based on validity in time domain and value domain to ensure the QoS (Quality of Service) in mobile and real-time database systems. The system will self-adjust with the feedback control scheduling algorithm to decrease damage. The algorithms and feedback control architecture can guarantee the QoS performance will not be beyond the reference by database administrator through simulation experiments.

Keywords Derived data, Quality of service, Real-time and mobile database, Update transaction

1 引言

近来移动实时数据服务在电子商务、股票交易、机械制造等行业应用越来越广泛。它不同于传统数据,实时数据有其时间语义,只有在对应的时间间隔内值才是有效的。为定义实时数据的正确性,首次引入时间有效性的概念[1-3]。因此当数据库中一个实时数据值真实反映了对应实体对象在物理环境中状态时,这个数据在时间域是新鲜的。高质量的需求以及严格的时间或数据新鲜度限制,实时数据库可能会出现负载超载,其结果使得很多事务错过截止期并违反数据新鲜度的限制。有效平衡用户事务和更新负载是提供满意服务的关键,这样才有可能满足用户指定的事务错过截止期比率和数据新鲜度。为实现这个目标,提出来很多技术如服务质量(QoS, Quality of Service),其中 K. D Kang 提出一种利用反馈控制名叫 QMF 方法[4]去保证事务满足截止期的比率和数据新鲜度。Emna Bouazizi 提出数据版本[5]的概念去解决事务间访问数据的冲突和加强并发控制。Lam Kam-Yiu 和 Xiong Ming 提出一种名为统计服务质量(SQoS, Statistical Quality of Service)[6]的技术,并采用 more-less 的方法控制系统中的数据新鲜度。而 Cosmin Rusu 和 Chen Jian-Jia 提出一种基于能耗的调度实时事务方法[7,8],对移动实时数据库系统控制能耗将有很大的用处。目前学者们提出了考虑事务错过截止期比率和数据新鲜度的概念,但他们考虑服务质量的影响因素都是孤立的。而且他们所提出的服务质量概念并没有涉及数据库系统中数据间的联系,即认为数据与数据间是相互孤立的,而实际上数据库系统中很多数据都是导出数据(Derived data),这些导出数据是由基本数据或其他导出数据构成的。

2 数据和事务模型

2.1 数据模型

本文讨论的数据库模型把数据对象分为时序数据和非时序数据两种。一个时序数据对象为反映现实世界的真实状态会持续改变(如股票的价格),而非时序对象随着时间的迁移

*)本文工作受青年国家自然科学基金项目(编号:60203017),留学回国人员基金及湖北省杰出青年基金的资助。向 军 讲师,博士研究生;李国徽 博士,教授,博士生导师。

不会变得过时(如身份证编号)。实时数据不同于传统数据,就是因为它们有时间语义。采样数据只有在某个时间间隔内才是有效的,这个时间间隔被称为数据的有效间隔。因此,当一个实时数据对象的值能真实反映该对象在物理环境下的状态时,认为这个数据对象是新鲜的。时序数据按照数据对象间的关系分为基本数据和导出数据两类。所谓基本数据是指该数据对象的值不依赖数据库中其它数据对象,而导出数据的值是靠基本数据或其它导出数据的值计算得来的,故提出读数据集(ReadSet)概念。当导出数据对象 d 来源于基本数据或者其它导出数据时,把构成 d 的数据元素的集合称为 d 的读数据集 ReadSet,简称为 $R(d)$。为描述数据库中数据对象间的关系,提出用一种有向非循环图(DAG,Directed Acyclic Graph)$G=(V,E)$来表示数据对象间的依赖关系,其中 V 表示图中的节点,E 表示节点间的关系。在 DAG 图中,一个节点代表一个数据对象,一条有向边代表读数据集中的一个元素,节点 b 表示基本数据,节点 d 表示导出数据。从图 1 中可以得出 $d4$ 的 ReadSet 为 $R(d4)=\{b1,b2,b3,d1,d2\}$。为更好遍历图中节点,每个节点被指定一个 level,其中基本数据节点 level(b)=1、导出数据节点的 level 定义为:level(d) = max{level(r),$r\in R(d)$} + 1。图 1 中相关节点的 level 为:

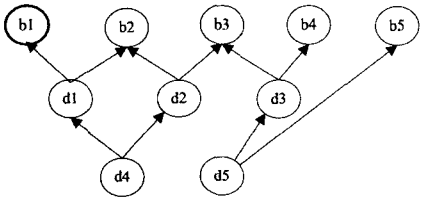

图 1 有向非循环图

$$\begin{aligned}\text{level}(b1) &= \text{level}(b2)\\ &= \text{level}(b3)\\ &= \text{level}(b4)\\ &= \text{level}(b5)\\ &= 1 \qquad (1)\\ \text{level}(d1) &= \text{level}(d2)\\ &= \text{level}(d3)\\ &= 1+1\\ &= 2 \qquad (2)\\ \text{level}(d4) &= \text{level}(d5)\\ &= 2+1\\ &= 3 \qquad (3)\end{aligned}$$

2.2 事务模型

根据事务的功能事务模型分为以下两类:更新事务和用户事务。更新事务的特点是:周期性更新时序数据对象,其目的是使得数据中时序对象的值真实反映物理世界数据对象的变化,而用户事务多用于读写非时序数据,而且用户事务到达是非周期性的。那些发起更新时序数据对象的更新事务称之为触发更新事务(Triggered Update Transaction),如图 1 中 $d4$ 需要被更新时,相对于 $R(d4)$的更新事务就是触发更新事务。

2.3 更新策略

立即更新(Immediate update)和按要求更新(On-demand update) 是主要讨论的两种更新策略,所谓立即更新是指一旦一个时序数据对象新的值可用时就马上更新数据库中时序对象的值,而按要求更新只有当数据库系统有相关进入系统的事务需要用到某一个时序数据对象时才更新数据库中的值。当系统负载较低时,最好选择立即更新,反之按要求更新表现出更好的性能。ODDFT 算法[9]是按要求更新中常用的遍历算法,并且 ODDFT 算法和普通算法比较能更好地改变更新频率去降低系统的能耗。为更好控制数据库综合 QoS 性能,利用反馈控制的调度方法改进 ODDFT,修改后的算法称为 MODDFT (Modifing On-Demand Depth-First Traversal)算法,详细讨论将在本文第 4 节进行。

3 QoS 规范和性能尺度

尽管很多学者提出了实时数据库中有关 QoS 性能的衡量参数标准数据的新鲜度(Data Freshness,DF)、事务错过截止期的比率(Miss Ratio of Transactions,MR),但并没有考虑环境的移动性和数据对象间的关系,在此基础上提出新的参数标准,如 CPU 的利用率(CPU Utilization)、QoS 综合性能等,下面将分别讨论。

3.1 数据新鲜度

在移动实时数据库系统中,一个时序数据在它变成无效之前必须被传感器事务更新,也就是说系统必须在其有效间隔期监测该数据对象的状态变化。因此,把数据新鲜度定义为用有效时间间隔解决物理环境和数据库中数据不一致性问题的一个性能参数。如果一个时序数据项 X_i 满足不等式(4),认为 X_i 是新鲜的或者说满足时序一致性。

$$current\_time - timestamp(X_i) \leqslant a_i \qquad (4)$$

不等式(4)中,current_time 表示物理世界的当前时间,a_i 表示 X_i 的绝对时间有效间隔。在实时数据库系统中,保证时序数据的新鲜是一个重要的设计目标。在最小化系统负载的前提下传统的方法有 one-one 和 half-half 等。但是,one-one 方法可能会使得一些时序数据变得无效,因为一个事务的连续两个子事务的分离可能会超过数据的有效间隔期。Half-half 方法尽管能保证数据的新鲜度,但和 one-one 方法相比,CPU 的负载至少翻倍。Xiong Ming 提出一种相对于前两种方法既能减小系统负载同时又能保证一定的数据新鲜度,名叫 more-less 方法[10]。通过分析发现以上三种方法都是针对周期性的事务,对于非周期性事务却没有涉及到。假设对数据项的更新频率是可以变化的,所以前面所提到的方法就存在诸多不足。

3.2 数据有效性

关于时序一致性的讨论,大多数研究人员都是在时间域内,系统为更新相关数据总是不惜代价。其实,下一个周期所取得的值虽然是新鲜的,但可能和数据库中的数据从值域来说并没有太大差别,甚至这个细微差别是系统可以容忍的。用 δ_x^t 表示为导出数据的 ReadSet 中元素 x 最大变化有效范围,$v_x^{t_0}$,v_x^t 分别表示为 x 在 t_0 和 t 时刻的值,当$|v_x^{t_0}-v_x^t|\leqslant\delta_x^t$ 成立时,导出数据 d 在值域内是新鲜的。

3.3 事务错过截止期比率

由于移动实时数据库本身潜在的局限,如网络连接中断、资源有限等可能会造成访问数据的时间延迟,影响数据的新鲜度。对于移动实时的事务,满足其截止期比数据的绝对正确性更为重要,讨论 QoS 的性能参数,其中之一就是满足事务截止期的比率。N_{tardy}表示错过截止期的事务数量,N 表示访问数据的事务总数,定义错过截止期的事务比率为:

$$MR = \frac{N_{tardy}}{N} \times 100\% \quad (5)$$

MR 作为 QoS 性能参数，由于数据库负载不可预测以及事务的访问模式的动态变化，有的事务错过截止期无法避免，但数据可以设定一个 MR 门槛值(如 2%)，当系统 $MR>2\%$ 时，必须采取措施降低 MR。同时为保证 QoS 性能，只控制 MR 是不够的，因为在一个时间段内可能访问数据的事务数量特别大，还有必要增加访问控制。

3.4 CPU 利用率

CPU 是移动实时数据库中最重要的系统资源，因此研究人员经常把降低 CPU 利用率同时满足一定的事务错过截止期比率(MR)和数据新鲜度(DF)作为系统设计目标。假定传感器事务都是周期性的，提出一些后面即将要用到的标记符号，定义如表 1。

表 1 算法中符号标记

标记	定义
X_i	时序数据 i
b_j	基本数据 j
d_k	导出数据 k
a_i	X_i 有效时间间隔
C_i	事务 t_i 的计算时间
p_i	事务 t_i 周期
D_i	事务 t_i 的相对截止期
t_i	事务 t_i
$t_i \rightarrow t_j$	事务 t_i 和事务 t_j 的先后执行顺序
$U_{i,j}$	当采用 $t_i \rightarrow t_j$ 执行顺序时，系统的 CPU 利用率 $U_{i,j} = \frac{c_i}{p_i} + \frac{c_j}{p_j}$

假定一个具有有效时间间隔为 a_i 的时序对象 X_i，事务 t_i 周期性更新 X_i，采用 more-less 更新策略，$p_i = \frac{N-1}{N}a_i$，$D_i = \frac{1}{N}a_i$，$C_i = \frac{1}{5}a_i$，则 $U_m = \frac{NC_i}{(N-1)a_i} = \frac{N}{5(N-1)}(N>1)$。假定系统有事务集 $\Gamma = \{\tau_i\}_{i=1}^{m}$，采用 DS-FP[11] 更新策略，系统的 CPU 利用率可以按式(6)计算，式(6)中 U 表示 CPU 的利用率，P_j 表示事务 t_j 的平均周期。

$$U = \sum_{i=1}^{m}\left(\frac{C_i}{v_i - \frac{C_i}{1-\sum_{j=1}^{i-1}\frac{C_j}{P_j}}}\right) \quad (6)$$

3.5 QoS 综合性能

实时数据库中，QoS 性能参数(如错过事务截止期的比率和数据的新鲜度)都是独立地衡量 QoS 性能。以前的研究人员并没有考虑这些参数之间的联系，实际上它们之间是相互联系的，所以一味追究某一个性能的高要求是没有意义的。构造一个 QoS 综合性能的函数如下：

$$\begin{aligned} QoS &= x \cdot DF + y \cdot MR + z \cdot U \\ &= x \cdot DF + y \cdot \frac{N_{tardy}}{N} \\ &\quad + z \cdot \sum_{i=1}^{m}\left(\frac{C_i}{v_i - \frac{C_i}{1-\sum_{j=1}^{i-1}\frac{C_j}{P_j}}}\right) \end{aligned} \quad (7)$$

式(7)中 DF，MR，U 分别表示 QoS 性能参数数据新鲜度、错过截止期事务比率、CPU 利用率，x、y、z 分别表示三个参数在 QoS 性能中所占的比例，数据管理员可以通过调整 x，y，z 平衡系统负载和 QoS 综合性能。

4 系统结构和事务调度算法

分布式移动实时数据库系统通常包含多个节点，类型分为移动支持基站(Mobile support station，MSS)和固定主机(Fixed host，FH)，系统中的所有节点都拥有自己的数据库，图 2 就是一个分布式移动实时数据库系统的系统架构图。本文中讨论的某个导出数据项就可能来源于系统中的多个节点。

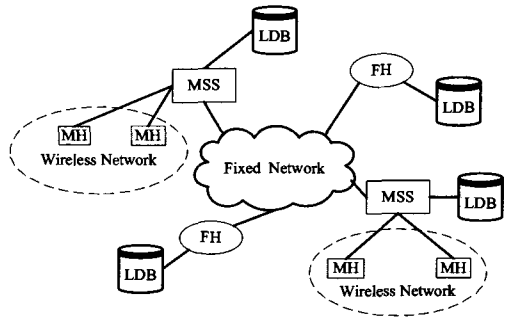

图 2 分布式移动数据库框架图

4.1 体系结构

利用反馈控制的体系结构控制系统事务错过截止期比率、数据更新度等 QoS 系统参数，3 个反馈控制用来保证系统的 QoS 及避免控制系统性能的偏差，使系统的性能更加接近由管理员说明的 QoS 规范。3 个循环控制分别见图 3。图 3 中每个控制循环都会产生一个控制信号。通过计算 QoS 说明参数和实际监测到信号的差别，然后基于反馈控制理论进行反馈控制，以达到使系统所监控的 3 个参数(数据新鲜度、CPU 利用率、事务错过截止期的比率)在管理员说明的范围之内，从而保证系统的 QoS。

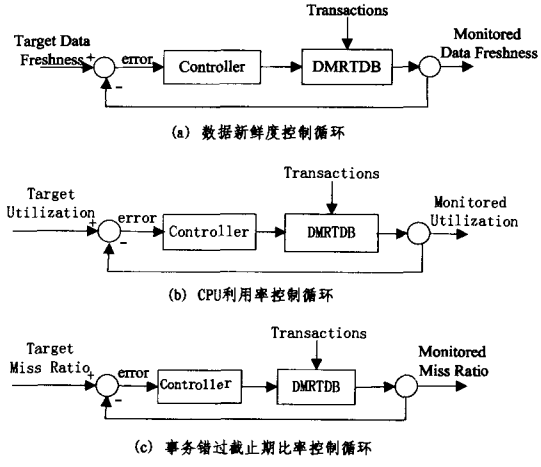

图 3 控制循环

4.2 更新事务的调度算法

系统反馈控制器中用 MODDFT 调度算法控制更新事务错过截止期的比率，算法定义一个函数 $error$ 表示某个数据项在某个时刻变化的最大值，定义如下：

$$error(x,t) = c * \log(t - timestamp(x)) \quad (8)$$

式(8)中 c 是一个常数，x 表示数据项，t 表示时间。如果数据项 x 在时刻 t 的错误小于 $error(x,t)$，算法认为这个错误是可以接受的，系统则不更新数据项 x。现假定有导出数据项 d，$R(d)$ 是 d 的 ReadSet，x 是 $R(d)$ 中的一个成员。为更新导出数据项 d，首先必须获得 $R(d)$ 的所有成员，提出一个获

取 Readset 的递归算法如图 4 所示。算法 4 中主要是通过判断基于图 1 中各个节点的 level 值和数据项 d 与队列 Q 中元素间的关系获得 $R(d)$。当更新某数据项时，所有与数据项更新有关联的触发更新事务进行了优先级的排序，更新的数据项越重要，则该触发更新事务就具有更高的优先级，优先级分派算法如图 5 所示。

```
    FindReadSet(d,Q)
t 表示更新数据项 d 的事务
Q 表示由 DAG 生成的队列
w 表示 Q 的一个元素
if level(d) = 1 then
        R(d) = ϕ
    d is a base data item
else
    for all w∈Q do
    if level(w) = level(d) − 1 and d→w then
        put w in R(d)
        put R(w) in R(d)
        FindReadset(w,Q)
    else if level(w) = 1 then
        put w in R(d)
          end if
        end for
end if
return R(d)
```

图 4　获取数据读集算法

$$total\_error = total\_error + error(x, freshness\_deadline) \tag{9}$$

式(9)用来计算更新数据项 x 时产生的错误总和，*freshness_deadline* 表示时间，当更新数据项 x 时，在某个时刻 $R(x)$的所有成员都是新鲜的，这个时刻就是 *freshness_deadline*。

```
AssignPriority(d,freshness_deadline)
  x 表示 R(d)的元素
for all x ∈R(d) do
  if error(x,freshness_deadline) ≥δ_x^t then
    total_error = total_error+
        error(x,freshness_deadline)
      put x in queue Q1
    end if
  end for
  for all x ∈Q1 do
    prio(x) = error(x,freshness_deadline) / total_error
  put x in queue Q2 stored by priority
end for
return Q2
```

图 5　触发事务优先级分派算法

基于 DAG 图调度更新事务的算法主要分两种：一种是深度优先遍历按要求更新算法(ODDFT)。另外一种是广度优先遍历按要求更新算法(ODBFT)，根据实验证明这两种算法在数据访问频繁的场合都有很高事务错过截止期比率，因为在产生更新事务时大量用户事务被延迟，导致错过截止期。而且在两个算法中，它们都孤立考虑服务质量的几个影响因子。提出综合考虑服务质量综合性能基础上，用改进后的 MODDFT 递归算法去调度系统中的更新事务，MODDFT 算法如图 6 所示。图 6 中 WCET(t)表示最坏场合的执行时间，P%表示 QoS 三个参数所占比例的门槛值，只要任意一个参数超过门槛值。算法就必须调整对应的影响因子 x(或 y,z)。算法中 *freshness_deadline* 被设定为启动更新事务的截止期算法用到了两种队列：一个是按优先级从高到低排列的数据项队列 $Q1$，另外一个是按释放时间排列的事务调度队列，而且 $Q1$ 清除了重复数据项。算法始终比较由监测到的 $MR(k)$,$U(k)$,$DF(k)$计算而来的 $QoS(k)$与 QoS_R 的大小，不断调整对应的影响因子。为提高算法效率，算法从值域比较了需要被更新的数据项，只有 $|v_x^{t0}-v_x^t|>\delta_x^t$ 的 $R(d)$的元素 x 才把对应的更新事务放入调度队列中。

```
    MODDFT(τ,freshness_deadline,schedule)
监测 MR(k),U(k),and DF(k),计算 QoS(k)
τ 表示更新数据项 d 的事务
Q1 表示 R(d)按优先级排序后的队列
        R(d) = FindReadSet(d)
Remove the duplicates of elements in R(d)
Q1 = AssignPriority(d,freshness_deadline)
Release_time(t) = deadline(t) − WCET(t)
Put t first in schedule
if QoS(k)> QoS_R then
        if MR(k) > MR_R * (1+p%)
            downgrade X
            recalculate QoS(k)
        else if U(k) > U_R * (1+p%)
            downgrade Y
            recalculate QoS(k)
        else DF(k) * (1+p%) < DF_R
            downgrade Z
            recalculate QoS(k)
        end if
end if
for all data items x in Q1 do
    if | v_x^0 − v_x^t |> δ_x^t then
      Let t_x be the transaction that updates x
      Remove duplicates of t_x from schedule
      Deadline(t_x) = release_time(t)
      MODDFT(t_x,freshness_deadline,schedule)
    else
      discard t_x
      end if
end for
```

图 6　更新事务调度算法 MODDFT

5　性能评估

5.1　仿真建立

在仿真中系统的负载主要是更新事务和用户事务，更新事务用来更新时序数据项，用户事务是指读取数据项和对数据项做逻辑或算术运算的事务。仿真中有两个参数可以动态调整：负载(load)和数据有效间隔值 δ_x^t。仿真就是通过给定不同的 load 和 δ_x^t 比较提出的算法是否能达到控制系统的 QoS 性能指标。QoS 性能主要包括稳定性能和暂态性能，稳定性能很多参数指标并不足以说明系统的性能是很好的，有可能出现某个参数瞬时的急剧变化给系统产生重大的损失。仿真具体环境设置如下：更新事务占整个负载的 50%，每次仿真运行时间约 10min，仿真实验用到的其他参数见表 2。

表 2 仿真实验参数

参数	值
更新事务数量	100
用户事务数量	100
δ_x^t	0.01
c	0.5
数据项个数	100
导出数据比例	0.5
基本数据比例	0.5
P%	1.5%

5.2　稳定性能

在稳定性能中平均事务错过截止期比率、CPU 利用率和数据新鲜度是主要的监测目标。为比较 MODDFT 算法，借用一些其他基本线来比较，如 NFC(没有用反馈控制)。在这个仿真中，实验主要是通过改变系统的负载看不同算法表现出的性能参数变化。做仿真前，数据管理员可以事先设定本系统的 QoS 规范，如 $MR_R=10\%$，$DF_R=90\%$，$U_R=85\%$，$x=1/3$，$y=1/3$，$z=1/3$。

根据仿真实验可以观察到平均事务错过截止期比率的变化，如图 7 所示。图 7 中 NFC 在系统超载时，MR 超过了

MR_R，而 MODDFT 即使在系统负载达到 200%时也没出现这种情况。MODDFT 算法能够保证这样效果的原因就是算法对系统监测信号采用反馈控制，而且对 MR 在整个 QoS 性能占的比例也做了限制。当然当系统过度的超载，即使 MODDFT 算法也有可能出现 NFC 算法的情况。

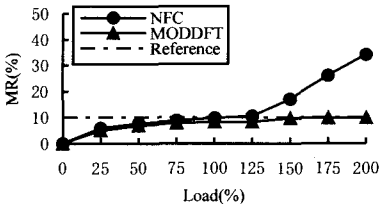

图 7　不同负载下平均 MR 的变化

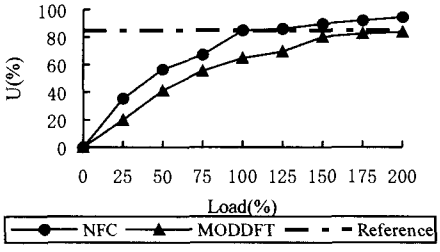

图 8　不同负载下平均 CPU 利用率的变化

同样，仿真中数据新鲜度和 CPU 的利用率 MODDFT 算法比 NFC 算法表现出了更好的性能，特别是在超载的环境下，平均 CPU 利用率在不同负载下变化的曲线图如图 8 所示，MODDFT 算法中的 CPU 利用率在超载 100%时仍在数据库管理员指定的 QoS 规范范围之内。

5.3　暂态性能

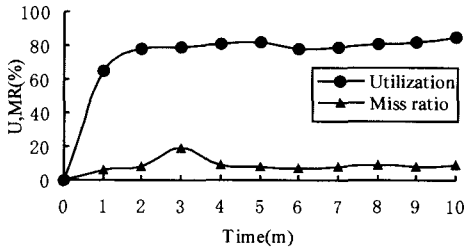

图 9　CPU 利用率和 MR 瞬时性能

稳态环境下的 QoS 性能保证对于负载不可预测的环境是必要的。但这还不够，因为有可能在瞬时出现过冲现象，所以在暂态性能环境本文主要观察系统有无过冲现象或者平衡解决时间。所谓平衡解决时间指的是系统从过冲到系统平衡所花的时间，而过冲现象一般是用来评估系统在最坏情况下性能变化。本仿真实验中主要观察 CPU 利用率、事务错过截止期比率的暂态，仿真结果如图 9 所示。从仿真开始到仿真结束均未出现过冲现象。

结束语　为控制数据服务，很多研究人员提出基于反馈控制结构的 QoS 的概念去保证一定的数据新鲜度、CPU 利用率和降低事务错过截止期比率。这些相关的工作都在文献[4,5,12,13]重点讨论了基本数据，但都没有考虑这些基本数据间的联系，而本文却考虑这一点。同时在值域提出数据有效性的方法去判断对应的数据项是否需要更新，如当数据项 d 满足 $|v_x^{t0} - v_x^t| = \delta_x^t$ 时，我们就认为 d 是新鲜的，没必要更新。所以，当更新一个导出数据时，考虑数据间的联系和数据值域有效性的问题可以大大减少更新事务的数量，从而提高 QoS。本文提出一种新的更新事务调度算法，而且从稳定性能和暂态性能两个方面去比较。该算法都能够表现出更好的性能，使得相关的性能指标在数据库管理员说明的规范范围之内。本文只是研究移动实时数据库 QoS 的一个开端，将来我们将继续把重点放在数据新鲜度和更新事务的调度算法，并尽量在实际应用中去实现。

参 考 文 献

[1] Locke D. Real-time Databases: Real-World Requirements, Real-time Database Systems: Issues and Applications[J]. Kluwer Academic Publishers, 1997: 83-91

[2] Ramamritham K. Real-time Databases[J]. Distributed and Parallel Databases, 1993, 1(2): 199-226

[3] Stankovic J A, Son S, Hansson J. Misconceptions About Real-Time Databases[J]. IEEE Computer, 1999, 32(6): 29-36

[4] Kang K D, Son S H, AStankovic J. A QoS-sensitive Approach for Timeliness and Freshness Guarantees in Real-time Databases// Proceedings of the 14th Euromicro Conference on Real-time Systems. Vienna, 2002

[5] Bouazizi E, Duvallet C, Sadeg B. Management of QoS and Data F-reshness in RTDBSs Using Feedback Control Scheduling and Data Versions // Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing. Seattle, 2005

[6] Lam K Y, Xiong Ming, Liang Bi Yu, et al. Statistical Quality of Service Guarantee for Temporal Consistency of Real-time Data Objects// Proceedings of the 25th IEEE International Real-Time Systems Symposium. Lisbon, 2004

[7] Rusu C, Ferreira A, Scordino C, et al. Energy-Efficient Real-Time Heterogeneous Server Clusters, Real-Time and Embedded Technology and Applications Symposium // Proceedings of the 12th IEEE. California, 2006

[8] Chen Jian-Jia, Hsu Heng-Ruey, Kuo Tei-Wei. Leakage-Aware Energy-Efficient Scheduling of Real-Time Tasks in Multiprocessor Systems, Real-time and Embedded Technology and Applications Symposium // Proceedings of the 12th IEEE. California, 2006

[9] Gustafsson T, Hansson J. Dynamic On-Demand Updating of Data in Real-time Database Systems // Proceedings of the 2004 ACM symposium on Applied computing. Nicosia, 2004

[10] Xiong Ming, Ramamritham K. Deriving Deadlines and Periods for Real-Time Update Transactions[J]. IEEE Transactions on Computers, 2004, 53(5): 567-583

[11] Xiong Ming, Han Song, Lam Kam-Yiu. A Deferrable Scheduling Algorithm for Real-Time Transactions Maintaining Data Freshness// Proceedings of the 26th IEEE International Real-Time Systems Symposium. Florida, 2005

[12] Lee V, Lam K, Son S H, et al. On Transaction Processing with Partial Validation and Timestamps Ordering in Mobile Broadcast Environments[J]. IEEE Transactions on Computers, Oct. 2002, 51(10): 1196-1211

[13] Gustafsson T, Hansson J. Data Management in Real-Time Systems: A Case of On-Demand Updates in Vehicle Control Systems// Proceedings of Real-Time and Embedded Technology and Applications Symposium. Toronto, 2004